J.-C. FANTOU

PRATIQUE des IBM PC et compatibles

GRAPHISME et SON

Volume 3

ÉDITIONS RADIO

PRATIQUE des IBM PC et compatibles

GRAPHISME et SON

J.-C. FANTOU

PRATIQUE des IBM PC et compatibles GRAPHISME et SON

Volume 3

ÉDITIONS RADIO

189, RUE SAINT-JACQUES - 75005 PARIS
TEL. (1).43.29.63.70

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal.

Éditions Radio - Siège social : 103, boulevard Saint-Michel - 75005 Paris

© Éditions Radio, Paris
1987

Tous droits de traduction, de reproduction et d'adaptation réservés pour tous pays.

Imprimé en France par Berger-Levrault, Nancy

Dépôt légal : juillet 1987
Éditeur n° 1060 - Imprimeur : 779021
I.S.B.N. 2 7091 1011 3

CHAPITRE I

LES MODES GRAPHIQUES CGA A MOYENNE ET HAUTE RÉSOLUTION

Les IBM PC et leurs compatibles mettent à votre disposition deux modes graphiques, l'un à moyenne résolution caractérisé par 320 × 200 points et 16 couleurs, l'autre à haute résolution comportant deux fois plus de points lumineux mais utilisables uniquement en deux couleurs, le blanc et le noir. De la maîtrise de ces quelques notions élémentaires dépend toute la suite que vous souhaitez apporter à la réalisation de vos graphiques personnalisés.

Menu du chapitre 1

Sélection des modes graphiques
Choix de la couleur de fond
Choix de la couleur d'encre
Allumage et extinction d'un point
Dessin d'un trait
Les coordonnées exprimées en valeurs relatives

Quel est votre choix : ■

SÉLECTION DES MODES GRAPHIQUES

Les IBM PC et leurs compatibles possèdent deux modes graphiques, l'un à moyenne résolution et l'autre à haute résolution. Le premier d'entre-eux, c'est-à-dire le mode graphique à moyenne résolution, est caractérisé par une résolution d'écran de 320 × 200 points lumineux pouvant être affichés en 16 couleurs différentes. Le second mode graphique, dit à haute résolution, possède quant à lui une résolution double du mode précédent, ce qui se traduit sur l'écran par une mosaïque de 640 × 200 points lumineux. Petite restriction toutefois : ces points ne peuvent s'afficher qu'en deux couleurs, le blanc et le noir !

SCREEN

Ceci dit, il est temps de passer aux actes !... La sélection du mode graphique à moyenne résolution s'obtient avec un...

Sélection du mode graphique à moyenne résolution... !

```
10 SCREEN 1
```

... tandis que celle du mode graphique à haute résolution nécessite un :

... ou du mode graphique à haute résolution !

```
10 SCREEN 2
```

Or, que constatez-vous après avoir exécutée l'une ou l'autre de ces deux instructions ?... Bien peu de choses en réalité !... L'écran affiche un **OK** au format 40 colonnes (appel du mode graphique à moyenne résolution) ou au format 80 colonnes (appel du mode graphique à haute résolution) suivi d'un curseur plein non clignotant à la ligne suivante dès l'exécution de l'instruction SCREEN(*).

C'est en fait la présence du curseur plein non clignotant qui vous signale que vous venez de quitter le mode texte (caractérisé rappelons-le par un curseur clignotant) pour aborder l'un des deux modes graphiques disponibles. Pour le reste, rien n'est changé !... Le texte **OK** et le curseur fixe apparaissent en blanc sur fond noir ainsi que la liste des touches de fonction.

Or, cette liste des touches de fonction n'a rien à faire sur un écran graphique !... Dans ces conditions, il convient donc de la supprimer en plaçant un...

Pour supprimer l'affichage des touches de fonction

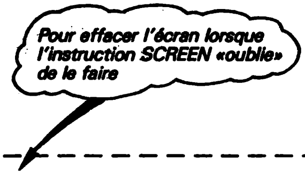
```
20 KEY OFF
```

... à la suite de l'instruction SCREEN.

Ce point étant acquis, il convient d'en aborder un autre, à savoir l'effacement de l'écran et le repositionnement du texte OK dans le coin supérieur gauche. Inutile, me direz-vous, car l'exécution de l'instruction SCREEN effectue implicitement cette double opération !... Or, cela n'est vrai que dans deux cas sur trois, à savoir lorsque vous lancez l'ordre SCREEN à partir d'un mode qui est différent de celui vers lequel vous vous dirigez. Dans le cas où les deux modes (origine et destinataire) sont identiques, comme par exemple lorsque vous tapez ou modifiez un programme graphique sous mode graphique, l'IBM PC n'efface pas l'écran (rien, ni personne n'est parfait...) et le texte **OK** apparaît à la suite du texte existant.

(*) La fiche détaillée de l'instruction SCREEN figure à la page 336 du volume 2 de Pratique des IBM-PC et compatibles.

Face à une telle situation, il convient donc de rajouter systématiquement un...




```
30 CLS
```

... à la suite des deux instructions précédente, de façon à éviter toute surprise désagréable.

Remarque : *Le retour au mode texte s'obtient grâce à un SCREEN 0, complété le cas échéant par un WIDTH 40 ou un WIDTH 80 selon le format souhaité.*

La sélection du mode graphique à moyenne résolution s'obtient avec SCREEN 1 et celle du mode graphique à haute résolution avec SCREEN 2.

CHOIX DE LA COULEUR DE FOND

On vous a laissé entendre en début de chapitre que le mode graphique à moyenne résolution disposait de 16 couleurs. Or, ceci n'est vrai qu'à 50 % pour les couleurs de fond et à 25 % pour les couleurs d'encre.
En ce qui concerne les couleurs de fond, celles-ci se répartissent en deux groupes de 8, l'un étant caractérisé par une saturation à 100 % des couleurs et l'autre par une saturation variable que vous pouvez doser à l'aide du bouton marqué d'une demi-lune de votre moniteur couleur.

Numéro de code	Couleur de fond en demi-teinte
0	Noir
1	Bleu
2	Vert
3	Turquoise
4	Rouge
5	Violet
6	Jaune
7	Blanc

Numéro de code	Couleur de fond saturé à 100 %
8	Noir
9	Bleu
10	Vert
11	Turquoise
12	Rouge
13	Violet
14	Jaune
15	Blanc

Comme vous pouvez le constater à la lumière de ce tableau, vous ne disposez en fait que de 8 couleurs différentes pour le fond d'écran et non de 16 comme vous eussiez pu le penser au départ, le noir (absence de couleur), les 3 couleurs primaires du moniteur (le rouge, le vert et le bleu), leur mélange deux à deux (vert et bleu pour le turquoise, rouge et bleu pour le violet et rouge et vert pour le jaune) et enfin le blanc, somme des 3 couleurs primaires.

COLOR

Le choix de la couleur de fond s'effectue à l'aide de l'instruction **COLOR(*)** qu'il suffit de faire suivre d'un numéro de code. Ainsi :

Le numéro 9 correspond au code de la couleur de fond bleue

```
40 COLOR 9
```

... a pour effet de rendre le fond d'écran tout bleu. Voici d'ailleurs à ce sujet un petit programme qui va vous permettre de visualiser les 8 couleurs de fond disponibles en teintes saturées et en demi-teintes :

Couleurs saturées (ligne 50)

Couleurs en demi-teinte (ligne 70)

```
10 SCREEN 1
20 CLS
30 KEY OFF
40 FOR C=0 TO 7
50 COLOR C+8
60 FOR T=1 TO 1000:NEXT
70 COLOR C
80 FOR T=1 TO 1000:NEXT
90 NEXT
100 COLOR 8
```

Celle-ci, c'est pour revoir le curseur ! (ligne 100)

Normalement si votre moniteur est bien réglé, vous devriez obtenir la suite des 8 couleurs de fond allant du noir au blanc en débutant pour chacune d'elle par la teinte vive (couleur saturée) suivie ensuite par la demi-teinte. Si tel n'est pas le cas, il ne vous reste plus qu'à reprendre les réglages de luminosité et de saturation couleur de votre moniteur.

Comment procéder à ces réglages ?... C'est tout simple !... Il vous suffit d'afficher le noir saturé avec un **COLOR 8** et de tourner ensuite le bouton de luminosité jusqu'à ce que le fond de l'écran cesse d'être obscur. Vous obtenez alors le réglage optimal de la luminosité. Ceci fait, il vous suffit de lancer le programme précédent et de doser vos demi-teintes à l'aide du bouton de saturation. Un point c'est tout !

(*) La fiche détaillée de l'instruction COLOR figure à la page 307 du volume 2 de Pratique des IBM-PC et compatibles.

Remarques : *En cas d'absence de numéro de code à la suite de l'instruction COLOR, le Basic maintient la dernière couleur de fond sélectionnée.*

A l'appel du mode graphique à moyenne résolution à l'aide de l'instruction SCREEN 1, c'est la couleur de fond n° 8 qui est sélectionnée par défaut.

L'instruction COLOR ne doit pas être employée avec le mode graphique à haute résolution qui est uniquement noir et blanc. Elle ne donnerait lieu qu'au message d'erreur « ILLEGAL FUNCTION CALL ».

Le mode graphique à moyenne résolution dispose de 8 couleurs de fond sélectionnables à l'aide de l'instruction COLOR.

CHOIX DE LA COULEUR D'ENCRE

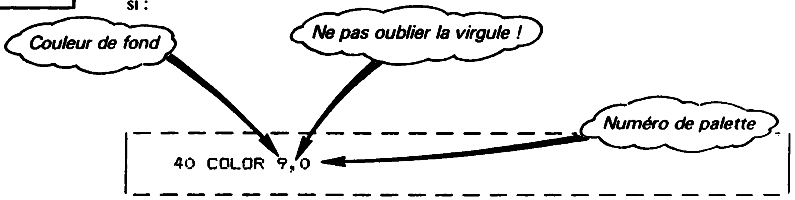
Et maintenant, si l'on parlait un peu des couleurs d'encre. De combien de couleurs différentes disposons-nous ?... Seize, huit ou quatre ?... En fait, vous ne disposez que de 8 couleurs réparties en deux palettes de 4, tel que le montre le tableau suivant :

Numéro de code	Palette n° 0	Palette n° 1
0	Identique à la couleur de fond	
1	Vert	Turquoise
2	Rouge	Violet
3	Jaune	Blanc

Comme sur un écran graphique on ne peut employer qu'une palette à la fois, il s'en suit que le choix des couleurs d'encre se limite à 4 pour une application donnée. Vous noterez au passage que, quelle que soit la palette choisie, la couleur n°0 correspond à la couleur de fond, ce qui permet l'effacement d'un dessin.

COLOR

Comment sélectionne-t-on une palette ?... Mais à l'aide encore une fois de l'instruction **COLOR** dont la syntaxe d'écriture veut que le numéro de palette soit indiqué à la suite de la couleur de fond en intercalant une virgule entre les deux numéros de code. Ainsi :



... sélectionne la palette n° 0 des couleurs d'encre avec le bleu (code n° 9) comme couleur de fond. De ce fait, vous allez disposer dans cet exemple précis de 4 couleurs d'encre : le bleu (code n° 0), le vert (code n° 1), le rouge (code n° 2) et le jaune (code n° 3).

Remarque : ***La palette n°1 est celle qui est sélectionée par défaut. Il n'est donc pas indispensable de préciser un numéro de palette à la suite de la couleur de fond si vous souhaitez travailler avec les couleurs d'encre de cette palette.***

Sélectionner une palette avec l'instruction **COLOR** c'est bien, mais cela ne nous dit toujours pas comment s'effectue le choix d'une couleur d'encre. Patience !... Il vous faut attendre pour cela les instructions de dessin, telles que **PSET, PRESET, LINE,** etc, etc...

Les 8 couleurs d'encre sont réparties en deux palettes (0 et 1) que l'on sélectionne à l'aide de l'instruction COLOR.

ALLUMAGE ET EXTINCTION D'UN POINT

La couleur de fond et la palette des couleurs d'encre étant sélectionnées, vous allez pouvoir commencer à dessiner quelques points lumineux sur l'écran. Pour cela, il vous faut savoir que l'écran graphique en moyenne résolution est constitué de 320 × 200 points disposés selon une matrice de 320 colonnes (numérotées de 0 à 319) et de 200 lignes (numérotées de 0 à 199). Vous noterez que la numérotation des colonnes s'effectue de gauche à droite tandis que celle des lignes s'effectue de haut en bas, contrairement à l'habitude.

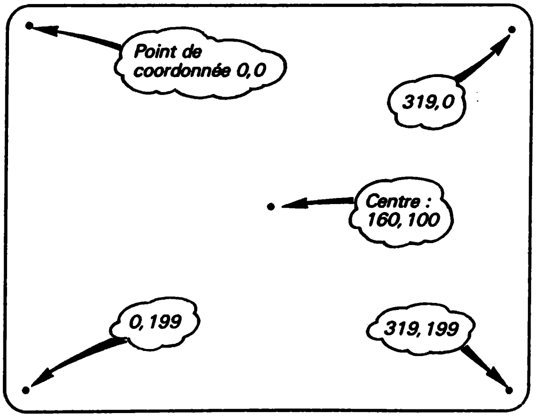


Fig. 1.1 : En moyenne résolution, l'écran graphique comporte 320 colonnes et 200 lignes

Vous noterez également que les coordonnées d'un point sont toujours désignées en commençant par le numéro de colonne, suivi ensuite du numéro de ligne.

En ce qui concerne le mode graphique à haute résolution, il en est presque de même à la différence près que les points lumineux sont deux fois plus nombreux et qu'ils sont disposés selon une matrice de 640 colonnes par 200 lignes, l'accroissement de définition se faisant exclusivement dans le sens horizontal, la définition verticale restant la même.

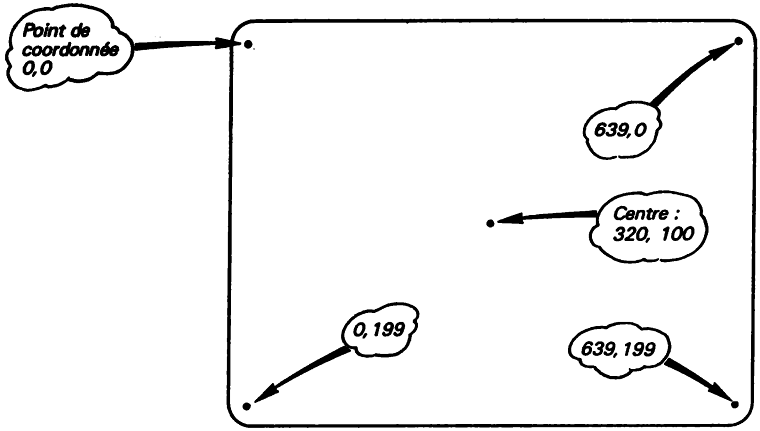


Fig. 1.2 : En haute résolution, l'écran graphique comporte 640 colonnes et 200 lignes.

Et maintenant, si l'on parlait un peu de l'instruction qui permet d'allumer ou d'éteindre un de ces points lumineux. En réalité, elles sont deux !... La première d'entre-elles, dénommée **PSET** sert à allumer un point lumineux, tandis que la seconde dénommée **PRESET** sert à l'éteindre.

PSET

Du point de vue syntaxe d'écriture, l'instruction **PSET** doit être obligatoirement suivie par les coordonnées du point à allumer placées entre parenthèses, suivies à leur tour par un numéro de couleur d'encre compris entre 1 et 3. C'est ainsi que pour allumer un point rouge au milieu de l'écran graphique à moyenne résolution, il vous faut écrire :

Numéro de code de la couleur rouge (palette n° 0)

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 0,0
40 PSET(160,100),2
```

Coordonnées du centre de l'écran

PRESET

Dans le même ordre d'idées, l'instruction **PRESET**(*) d'effacement d'un point lumineux nécessite, elle aussi, la présence des coordonnées du point placées entre parenthèses. Par contre, il est inutile d'indiquer la moindre couleur d'encre, celle-ci étant systématiquement la même que la couleur de fond de l'écran. A titre d'exemple, voici comment effacer le point précédent :

L'indication de la couleur d'encre n° 0 est inutile à la suite de l'instruction PRESET

```
50 PRESET (160,100)
```

Remarque : *Il est possible d'omettre également la couleur d'encre à la suite de l'instuction PSET. Dans ce cas, c'est la couleur d'encre n° 3 qui est sélectionnée par défaut (jaune avec la palette n° 0 ou blanc avec la palette n° 1).*

Pour illuster ce qui vient d'être dit, voici un petit programme qui va vous remplir l'écran de points lumineux de couleurs différentes et qui va les disposer selon le plus grand des hasards, à la manière d'un ciel étoilé :

Allumage d'un point de coordonnées X, Y et de couleur C (ligne 70)

Extinction d'un point de coordonnées X, Y (ligne 100)

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 0,0
40 X=INT(RND*320)
50 Y=INT(RND*200)
60 C=INT(RND*3)+1
70 PSET (X,Y),C
80 X=INT(RND*320)
90 Y=INT(RND*200)
100 PRESET (X,Y)
110 GOTO 40
```

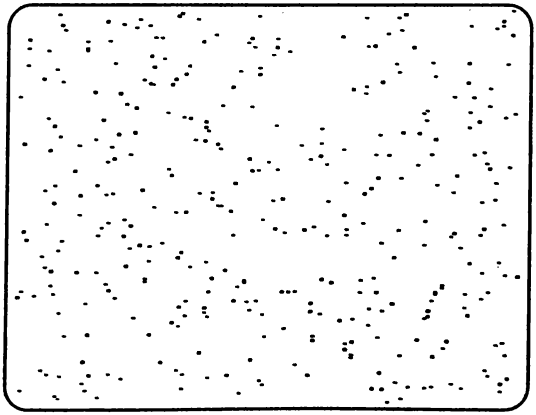


Fig. 1.3. : Ciel constellé d'étoiles obtenu en allumant et en éteignant au hasard un grand nombre de points.

Dessiner des points au hasard, c'est bien, mais cela ne mène pas très loin !... Voici un autre exemple d'application de l'instruction **PSET** (*) qui, bien qu'un peu long à écrire, va vous dessiner la carte de France sur votre écran :

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 8,1
40 FOR I=1 TO 677
50 READ X,Y
60 PSET (X,Y),3
70 NEXT
100 DATA 133,131,133,130,132,129,132,128,131,127
110 DATA 131,126,130,125,130,124,129,123,128,122
120 DATA 127,121,126,121,125,120,124,120,124,119
130 DATA 124,118,125,117,125,116,125,115,125,114
140 DATA 124,113,124,112,124,111,125,110,124,110
150 DATA 123,110,122,110,121,109,120,108,119,107
160 DATA 118,106,117,105,116,104,116,103,115,102
170 DATA 114,101,114,100,114,99,114,98,114,97
180 DATA 114,96,113,95,114,94,115,94,116,94
190 DATA 117,93,116,92,115,92,114,92,113,92
200 DATA 112,92,111,93,110,93,109,92,108,91
210 DATA 108,90,108,89,109,88,108,88,107,88
220 DATA 106,88,105,87,104,87,103,88,102,88
230 DATA 101,88,100,87,99,86,98,86,97,85
240 DATA 96,84,95,83,94,83,93,83,92,82
250 DATA 91,82,90,82,89,83,88,83,87,82
260 DATA 87,81,86,80,85,79,84,79,83,79
270 DATA 82,78,82,77,83,76,84,76,85,76
280 DATA 86,76,87,76,88,76,87,75,86,75
290 DATA 85,75,84,74,84,73,84,72,85,72
300 DATA 86,72,87,72,88,72,87,71,86,71
310 DATA 85,71,84,71,83,71,82,70,82,69
320 DATA 83,68,84,67,85,66,86,66,87,65
330 DATA 88,65,89,65,90,65,91,65,92,65
340 DATA 93,65,94,66,95,66,96,66,96,65
350 DATA 96,64,97,63,98,63,99,63,100,63
360 DATA 101,63,102,63,103,64,104,65,105,66
370 DATA 106,67,106,68,107,69,108,69,109,68
380 DATA 109,67,110,66,111,66,112,67,113,67
390 DATA 114,67,115,66,116,65,117,65,118,66
400 DATA 118,67,119,68,120,68,121,68,122,67
410 DATA 121,67,120,66,120,65,120,64,120,63
420 DATA 120,62,120,61,120,60,120,59,120,58
430 DATA 119,57,118,56,117,55,117,54,117,53
440 DATA 117,52,116,51,116,50,115,49,115,48
450 DATA 116,47,117,47,118,48,119,48,120,49
460 DATA 121,49,122,48,123,48,124,48,125,48
470 DATA 125,49,125,50,125,51,124,52,125,53
480 DATA 125,54,126,55,127,54,128,53,129,53
490 DATA 130,53,131,54,132,54,133,54,134,54
500 DATA 135,55,136,55,137,56,138,55,139,54
510 DATA 140,54,141,54,142,54,143,53,144,53
520 DATA 145,53,145,52,144,52,143,52,142,52
530 DATA 141,52,140,51,140,50,140,49,141,48
540 DATA 142,47,143,46,144,46,145,45,146,45
550 DATA 147,45,148,44,149,44,150,44,151,44
560 DATA 152,43,153,42,154,41,155,40,156,40
570 DATA 157,40,158,39,157,38,157,37,157,36
580 DATA 157,35,157,34,157,33,157,32,156,31
590 DATA 156,30,156,29,157,28,157,27,157,26
```

```
600 DATA 158,25,159,24,160,24,161,24,162,24
610 DATA 163,23,164,23,165,23,166,23,167,22
620 DATA 168,23,169,24,169,25,170,26,170,27
630 DATA 171,28,172,28,173,27,174,27,175,27
640 DATA 176,28,177,29,177,30,178,31,179,31
650 DATA 180,32,181,32,182,33,183,34,184,34
660 DATA 185,34,186,34,187,35,188,36,188,37
670 DATA 188,38,188,39,188,40,188,41,189,41
680 DATA 190,41,191,40,192,39,192,38,193,37
690 DATA 194,36,195,36,196,37,196,38,196,39
700 DATA 196,40,196,41,196,42,196,43,197,44
710 DATA 198,45,199,45,200,46,201,47,202,47
720 DATA 203,48,204,48,205,47,206,47,207,47
730 DATA 208,48,209,48,210,48,211,48,212,47
740 DATA 213,47,214,48,215,48,216,49,217,50
750 DATA 217,51,218,52,219,52,220,52,221,52
760 DATA 222,52,223,52,224,52,225,52,226,52
770 DATA 227,52,228,53,229,53,230,53,231,53
780 DATA 232,53,233,53,234,53,235,54,236,54
790 DATA 236,55,236,56,236,57,235,58,235,59
800 DATA 234,60,233,61,233,62,232,63,232,64
810 DATA 232,65,232,66,232,67,231,68,231,69
820 DATA 231,70,231,71,231,72,231,73,231,74
830 DATA 231,75,231,76,231,77,231,78,232,79
840 DATA 232,80,232,81,231,82,230,83,229,84
850 DATA 228,84,227,84,226,84,226,83,225,83
860 DATA 224,84,224,85,225,86,226,86,226,87
870 DATA 225,88,224,89,224,90,223,91,222,92
880 DATA 221,93,220,93,220,94,220,95,219,96
890 DATA 219,97,218,98,217,99,217,100,216,101
900 DATA 216,102,216,103,216,104,217,104,218,104
910 DATA 219,104,220,103,221,103,222,102,223,102
920 DATA 224,102,225,103,225,104,225,105,225,106
930 DATA 226,107,226,108,227,109,228,110,228,111
940 DATA 228,112,227,113,226,114,227,115,228,116
950 DATA 228,117,229,118,230,119,230,120,230,121
960 DATA 230,122,230,123,230,124,229,125,228,125
970 DATA 227,126,226,126,225,127,226,128,227,129
980 DATA 228,129,229,129,230,130,230,131,230,132
990 DATA 230,133,230,134,229,135,229,136,229,137
1000 DATA 230,138,230,139,230,140,231,141,232,141
1010 DATA 233,141,234,142,235,142,236,141,237,141
1020 DATA 238,141,239,141,240,142,240,143,240,144
1030 DATA 240,145,239,146,239,147,239,148,239,149
1040 DATA 238,150,237,150,236,151,235,152,234,153
1050 DATA 234,154,233,155,232,155,231,156,230,157
1060 DATA 229,157,228,158,229,159,229,160,229,161
1070 DATA 228,161,227,162,226,162,225,163,224,163
1080 DATA 223,163,223,164,222,165,221,166,220,165
1090 DATA 219,164,218,164,217,164,216,164,215,164
1100 DATA 214,163,213,163,212,163,211,163,210,162
1110 DATA 210,161,209,160,208,160,207,160,206,159
1120 DATA 205,159,204,160,203,159,202,159,202,160
1130 DATA 201,161,200,161,199,160,199,159,198,159
1140 DATA 197,159,196,159,195,159,194,158,194,157
1150 DATA 193,157,192,158,191,159,190,160,189,161
```

(*) Les fiches détaillées des instructions PSET et PRESET figurent aux pages 332 et 331 du volume 2 de Pratique des IBM-PC et compatibles.

```
1160 DATA 188,162,187,163,186,163,185,163,184,164
1170 DATA 183,165,182,166,182,167,182,168,182,169
1180 DATA 182,170,182,171,182,172,182,173,182,174
1190 DATA 183,175,183,176,184,177,183,178,182,179
1200 DATA 181,178,180,178,179,178,178,178,177,179
1210 DATA 176,179,175,179,174,179,173,179,172,179
1220 DATA 171,178,170,179,169,179,168,180,167,179
1230 DATA 166,179,165,178,164,178,163,177,164,176
1240 DATA 163,175,162,175,161,176,160,176,159,175
1250 DATA 158,174,157,174,156,173,155,173,154,172
1260 DATA 153,172,152,172,151,171,150,171,150,172
1270 DATA 150,173,150,174,149,174,148,174,147,174
1280 DATA 146,174,145,174,144,174,143,174,142,174
1290 DATA 141,174,140,174,139,173,138,172,137,172
1300 DATA 136,172,135,173,134,173,133,172,132,171
```

```
1310 DATA 131,170,130,170,129,169,128,169,127,169
1320 DATA 126,168,125,168,124,168,124,169,123,169
1330 DATA 122,169,121,168,122,167,122,166,122,165
1340 DATA 122,164,121,164,120,165,119,164,118,163
1350 DATA 117,162,118,161,119,160,120,159,121,158
1360 DATA 121,157,121,156,122,155,122,154,122,153
1370 DATA 122,152,122,151,123,150,123,149,123,148
1380 DATA 123,147,123,146,123,145,123,144,124,143
1390 DATA 124,142,124,141,125,140,126,140,127,139
1400 DATA 127,138,126,138,125,138,124,138,124,137
1410 DATA 124,136,124,135,124,134,124,133,124,132
1420 DATA 124,131,124,130,125,129,125,128,125,127
1430 DATA 125,126,125,125,125,124,126,124,127,124
1440 DATA 128,125,128,126,129,127,129,128,130,129
1450 DATA 131,130,132,131
```

Si vous avez eu le courage d'aller jusqu'au bout de cette longue liste de données, vous devirez aboutrir à ceci (à vos erreurs de frappe près...) :

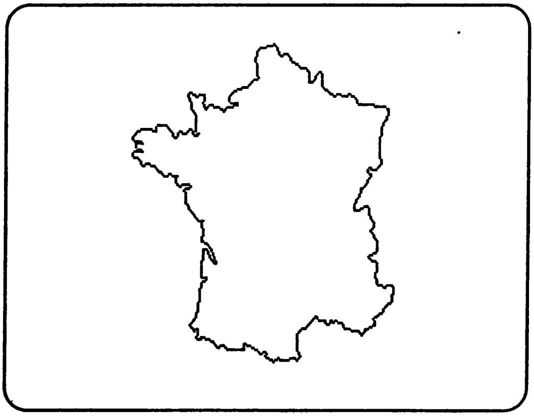


Fig. 1.4. : Carte de France dessinée avec quelques données et l'instruction PSET.

Au passage, vous noterez l'emploi d'une seule instruction PSET associée à une boucle de lecture de données employant les instructions READ et DATA. Cette disposition est nettement préférable lorsque le nombre de points à dessiner est important, comme c'est le cas ici.

L'allumage d'un point s'obtient avec l'instruction PSET et son extinction avec l'instruction PRESET.

DESSIN D'UN TRAIT

Dessiner un trait avec l'instruction **PSET** est une chose aisée. Il suffit pour cela de placer cette dernière dans une boucle. Ainsi, pour dessiner un trait horizontal en plein milieu de l'écran, vous n'auriez qu'à écrire :

Cette boucle dessine (lentement) un trait horizontal

```
10 KEY OFF
20 SCREEN 1 :CLS
30 COLOR 0,1
40 Y=100
50 FOR X=0 TO 319
60 PSET (X,Y),3
70 NEXT
```

LINE

Toutefois, une telle méthode est peu exploitable car elle donne lieu à des temps d'exécution trop longs (à moins, bien sûr, que cela soit recherché pour un effet artistique ou autre...). La solution (rapide, s'entend...) passe par l'emploi de l'instruction **LINE** (*) spécialement adaptée au tracé de traits, qu'ils soient horizontaux, verticaux ou obliques.

Sa syntaxe d'écriture, du même style que celle de PSET, nécessite que soient indiquées entre parenthèses les coordonnées du point de départ et du point d'arrivée, le tout étant clôturé par l'indication de la couleur d'encre du trait. Ainsi, pour dessiner le même trait que précédemment, il vous suffit d'écrire cette fois-ci :

Indication de la couleur du trait

```
10 KEY OFF
20 SCREEN 1 :CLS
30 COLOR 0,1
40 LINE(0,100)-(319,100),3
```

Coordonnées du point de départ

Coordonnées du point d'arrivée

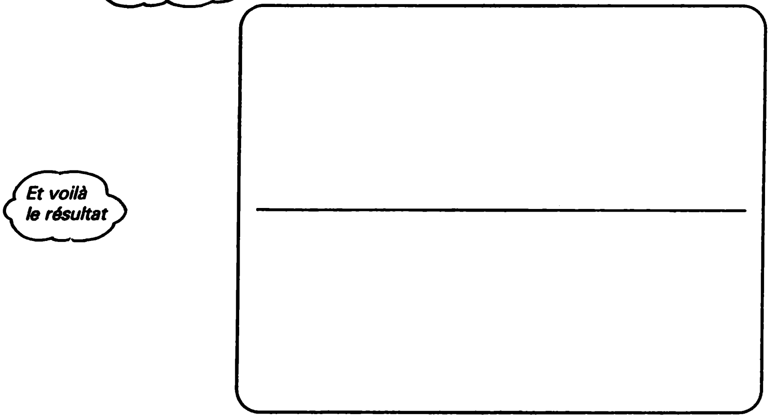


Fig. 1.5. : Dessin d'un trait horizontal en plein milieu de l'écran.

(*) La fiche détaillée de l'instruction LINE figure à la page 318 du volume 2 de Pratique des IBM-PC et compatibles.

Vous noterez l'emploi d'un trait d'union en tant que séparateur des coordonnées de départ et d'arrivée.

Remarque : *Comme précédemment, l'absence d'indication de couleur d'encre à la suite de l'instruction LINE sélectionne automatiquement la couleur n° 3 (jaune avec la palette n° 0 ou blanc avec la palette n° 1).*

Dessiner (rapidement) un trait, c'est bien, mais cela ne suffit pas. Vos graphiques en comportent nécessairement un grand nombre et l'on peut se poser la question de savoir s'il est indispensable de devoir redéfinir pour chacun d'eux les coordonnées des points de départ et d'arrivée ?... A cela, la réponse est non !... En effet, le Basic garde en mémoire les coordonnées d'arrivée du dernier point tracé, qu'il l'ait été avec l'instruction **PSET** ou avec l'instruction **LINE**.

Dans ces conditions, il vous suffit de n'indiquer que le point d'arrivée du trait, celui-ci étant dessiné à partir du précédent point d'arrivée gardé en mémoire, chaque fois que vous avez à dessiner une succession de traits s'enchaînant les uns à la suite des autres. A titre d'exemple de démonstration, voici comment former un cadre autour de l'écran en employant (autant que faire ce peut...) l'écriture simplifiée :

```
10 KEY OFF
20 SCREEN 1 :CLS
30 COLOR 0,1
40 LINE(0,0)-(319,0),3
50 LINE-(319,199),3
60 LINE-(0,199),3
70 LINE-(0,0),3
```

Ce programme dessine un cadre autour de l'écran

En écriture, simplifiée, ne figurent que les coordonnées du point d'arrivée

Vous remarquerez que, bien qu'employant l'écriture simplifiée de l'instruction **LINE** à trois reprises (lignes 50, 60 et 70), le programme nécessite l'écriture complète des coordonnées pour le premier trait du cadre (ligne 40) de façon à fixer en mémoire les coordonnées du dernier point tracé.

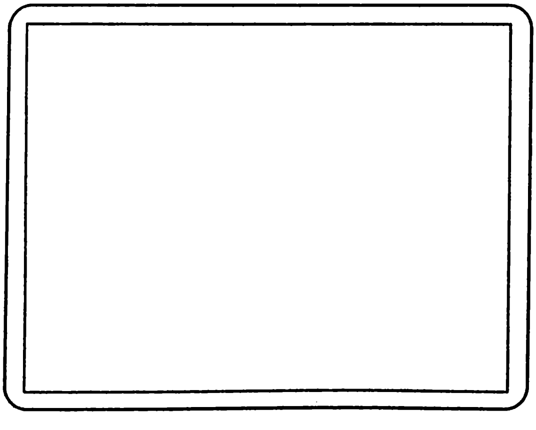


Fig. 1.6. : Cadre autour de l'écran obtenu avec l'instruction LINE.

Et les traits pointillés dans tout cela ?... L'instruction LINE est-elle capable de dessiner des traits pointillés ?... Mais oui !... Il suffit pour cela de placer deux virgules supplémentaires à la suite de l'instruction LINE et d'indiquer (en héxadécimal..) la valeur du code binaire à 16 éléments binaires qui définit les points qui devront être allumés de ceux qui devront être éteints.

Ainsi, si vous souhaitez redessiner le cadre précédent avec un trait pointillé constitué de deux points allumés suivis de deux points éteints (code binaire 1100 1100 1100 1100), il vous suffit de placer la valeur hexadécimale de ce code, soit CC CC à la suite de la 3ᵉ virgule, comme ceci :

```
10 KEY OFF
20 SCREEN 1 :CLS
30 COLOR 0,1
40 LINE(0,0)-(319,0),3,,&HCCCC
50 LINE-(319,199),3,,&HCCCC
60 LINE-(0,199),3,,&HCCCC
70 LINE-(0,0),3,,&HCCCC
```

Code hexadécima d'allumage et d'extinction des points du trait...

... qu'il faut placer à la suite de la 3ème virgule !

L'emploi de ce code d'allumage et d'extinction des points amène quelques commentaies. Tout d'abord, les « 1 » du code binaire correspondent aux points allumés et les « 0 » aux points éteints. Ensuite, ce code présent sous sa forme hexadécimale doit être précédé par les caractères &H. Enfin, la position de ce code à la suite de la 3ᵉ virgule de l'instruction LINE laisse supposer que celle-ci possède d'autres options qui n'ont pas encore été décrites (mais qui ne vont pas tarder à l'être !...).

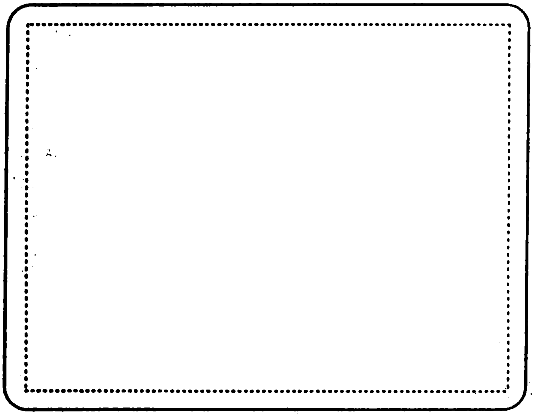


Fig. 1.7 : Traits en pointillés obtenus en indiquant à la suite de l'instruction LINE le code d'allumage et d'extinction des points.

Comme vous pouvez le constater, l'instruction LINE est une instruction puissante qui ouvre la porte à bon nombre d'applications graphiques. A titre d'exemple, en voici une qui va vous afficher en grand sur votre écran la marque de votre ordinateur préféré :

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 9,1
40 FOR Y=70 TO 126 STEP 8
50 FOR I=0 TO 3
60 LINE(0,Y+I)-(319,Y+I),3
70 NEXT I,Y
80 FOR I=1 TO 186
90 READ X1,Y1,X2,Y2
100 LINE(X1,Y1)-(X2,Y2),0
110 NEXT
120 DATA 0,70,84,70,0,71,84,71
130 DATA 0,72,84,72,0,73,84,73
140 DATA 0,78,84,78,0,79,84,79
150 DATA 0,80,84,80,0,81,84,81
160 DATA 0,86,90,86,0,87,90,87
170 DATA 0,88,90,88,0,89,90,89
180 DATA 0,94,90,94,0,95,90,95
190 DATA 0,96,90,96,0,97,90,97
200 DATA 0,102,90,102,0,103,90,103
210 DATA 0,104,90,104,0,105,90,105
220 DATA 0,110,90,110,0,111,90,111
230 DATA 0,112,90,112,0,113,90,113
240 DATA 0,118,84,118,0,119,84,119
250 DATA 0,120,84,120,0,121,84,121
260 DATA 0,126,84,126,0,127,84,127
270 DATA 0,128,84,128,0,129,84,129
280 DATA 110,70,115,70,110,71,115,71
290 DATA 110,72,115,72,110,73,115,73
300 DATA 110,78,115,78,110,79,115,79
310 DATA 110,80,115,80,110,81,115,81
320 DATA 104,86,121,86,104,87,121,87
330 DATA 104,88,121,88,104,89,121,89
340 DATA 104,94,121,94,104,95,121,95
350 DATA 104,96,121,96,104,97,121,97
360 DATA 104,102,121,102,104,103,121,103
370 DATA 104,104,121,104,104,105,121,105
380 DATA 104,110,121,110,104,111,121,111
390 DATA 104,112,121,112,104,113,121,113
400 DATA 110,118,115,118,110,119,115,119
410 DATA 110,120,115,120,110,121,115,121
420 DATA 110,126,115,126,110,127,115,127
430 DATA 110,128,115,128,110,129,115,129
440 DATA 135,86,147,86,135,87,147,87
450 DATA 135,88,147,88,135,89,147,89
460 DATA 135,110,147,110,135,111,147,111
470 DATA 135,112,147,112,135,113,147,113
480 DATA 151,70,166,70,153,71,166,71
490 DATA 155,72,166,72,156,73,166,73
500 DATA 160,78,166,78,160,79,166,79
510 DATA 161,80,166,80,161,81,166,81
520 DATA 161,86,172,86,161,87,172,87
530 DATA 161,88,172,88,160,89,172,89
540 DATA 158,94,172,94,157,95,172,95
550 DATA 156,96,172,96,155,97,172,97
560 DATA 155,102,172,102,156,103,172,103
570 DATA 157,104,172,104,158,105,172,105
580 DATA 160,110,172,110,161,111,172,111
590 DATA 161,112,172,112,161,113,172,113
600 DATA 161,118,166,118,161,119,166,119
610 DATA 160,120,166,120,160,121,166,121
620 DATA 156,126,166,126,155,127,166,127
630 DATA 153,128,166,128,151,129,166,129
640 DATA 186,70,215,70,186,71,215,71
650 DATA 187,72,214,72,187,73,214,73
660 DATA 190,78,211,78,190,79,211,79
670 DATA 191,80,210,80,191,81,210,81
680 DATA 194,86,207,86,194,87,207,87
690 DATA 195,88,206,88,195,89,206,89
700 DATA 198,94,203,94,198,95,203,95
710 DATA 199,96,202,96,199,97,202,97
720 DATA 186,102,187,102,186,103,187,103
730 DATA 186,104,188,104,186,105,188,105
740 DATA 186,110,191,110,186,111,191,111
750 DATA 186,112,192,112,186,113,192,113
760 DATA 186,118,194,118,186,119,195,119
770 DATA 186,120,195,120,186,121,196,121
780 DATA 186,126,198,126,186,127,198,127
790 DATA 186,128,199,128,186,129,199,129
800 DATA 214,102,215,102,214,103,215,103
810 DATA 213,104,215,104,213,105,215,105
820 DATA 210,110,215,110,210,111,215,111
830 DATA 209,112,215,112,209,113,215,113
840 DATA 207,118,215,118,206,119,215,119
850 DATA 206,120,215,120,205,121,215,121
860 DATA 203,126,215,126,203,127,215,127
870 DATA 202,128,215,128,202,129,215,129
880 DATA 235,70,319,70,235,71,319,71
890 DATA 235,72,319,72,235,73,319,73
900 DATA 235,78,319,78,235,79,319,79
910 DATA 235,80,319,80,235,81,319,81
920 DATA 229,86,319,86,229,87,319,87
930 DATA 229,88,319,88,229,89,319,89
940 DATA 229,94,319,94,229,95,319,95
950 DATA 229,96,319,96,229,97,319,97
960 DATA 229,102,319,102,229,103,319,103
970 DATA 229,104,319,104,229,105,319,105
980 DATA 229,108,319,108,229,109,319,109
990 DATA 229,110,319,110,229,111,319,111
1000 DATA 229,112,319,112,229,113,319,113
1010 DATA 235,118,319,118,235,119,319,119
1020 DATA 235,120,319,120,235,121,319,121
1030 DATA 235,126,319,126,235,127,319,127
1040 DATA 235,128,319,128,235,129,319,129
```

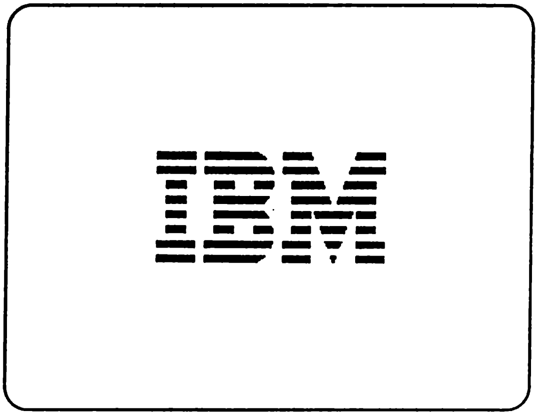


Fig. 1.8. : Sigle IBM bien connu obtenu avec « quelques » instructions LINE.

Vous vous demandez peut-être pourquoi il faut tant de DATA pour dessiner le sigle IBM ?... Ne cherchez pas !... C'est tout simplement qu'à travers cet exemple, l'auteur a recherché un effet visuel spectaculaire. Si vous voulez savoir lequel, il n'y a qu'une solution : taper le programme !

L'instruction LINE permet de dessiner des traits droits, qu'ils soient horizontaux, verticaux ou obliques.

LES COORDONNÉES EXPRIMÉES EN VALEURS RELATIVES

Comme Monsieur Jourdain qui faisait de la prose sans le savoir, vous voici en train de découvrir qu'à travers les quelques exemples que vous venez de taper, vous n'avez cessé d'exprimer les coordonnées de l'écran graphique en valeurs absolues, c'est-à-dire en indiquant les numéros de colonne et de ligne de chacun des points considérés.

Or, cette méthode n'est pas la seule qui puisse exister !... Vous pouvez tout aussi bien exprimer les coordonnées de chacun des points en n'indiquant que la différence existant par rapport au dernier point tracé. C'est cela, les coordonnées exprimées en valeurs relatives !

Mais en quoi l'expression des coordonnées en valeurs relatives peut-elle présenter un intérêt quelconque ?... Sur des images fixes, certainement aucun !... Par contre, sur des images animées, cela peut s'avérer très pratique car cela accélère notablement la vitesse d'exécution des lignes d'instructions.

Pour illustrer ce qui précède, voici un petit programme qui dessine une cible sur l'écran en un point définit par les variables X et Y. Ainsi, il suffit de faire varier la valeur de ces variables (à l'aide du clavier par exemple...) pour positionner la cible où bon vous semble :

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 0,0
40 X=160:Y=100
50 LINE(X-3,Y)-(X-2,Y),2
60 LINE(X+2,Y)-(X+3,Y),2
70 LINE(X,Y-3)-(X,Y-2),2
80 LINE(X,Y+2)-(X,Y+3),2
```

La position de cette cible est définie en valeurs absolues...

STEP

Cette première écriture employant la notation absolue, en voici une seconde qui réalise strictement le même dessin, mais cette fois-ci en notation relative. Notez la présence de l'instruction **STEP** (*) devant chaque ensemble de coordonnées :

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 0,0
40 X=160:Y=100
50 PRESET(X,Y)
60 LINE STEP(-3,0)-STEP(1,0),2
70 LINE STEP(4,0)-STEP(1,0),2
80 LINE STEP(-3,-3)-STEP(0,1),2
90 LINE STEP(0,4)-STEP(0,1),2
```

... tandis que celle-ci l'est en valeurs relatives !

Par rapport au programme précédent, vous pouvez constater qu'avec la notion relative, il faut *impérativement* fixer le point de coordonnées X, Y. C'est **PRESET** qui réalise cela dans le cas présent (ligne 50) puisque le centre de la cible est vide.

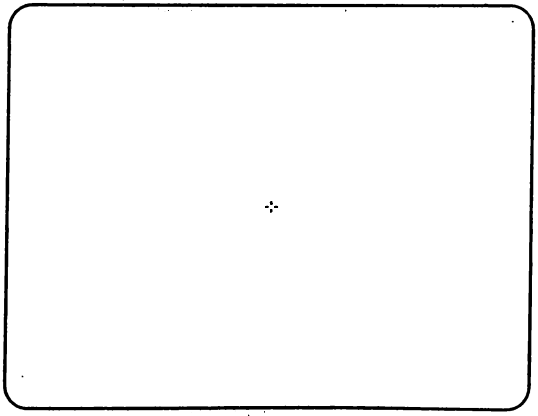


Fig. 1.9. : Cible dessinées à partir de coordonnées exprimées en valeurs absolues ou relatives

L'instruction STEP autorise la notation relative des coordonnées des points.

(*) La fiche détaillée de l'instruction STEP, figure à la page 339 du volume 2 de Pratique des IBM-PC.

Exercices sur le chapitre 1 :

1. De combien de points lumineux dispose le mode graphique à haute résolution :

☐ 320 × 200
☐ 640 × 200
☐ 640 × 400

2. Est-il nécessaire de rajouter une instruction CLS à la suite d'une instruction SCREEN ?

☐ Non, car l'instruction SCREEN efface systématiquement l'écran.
☐ Oui, pour pallier au cas où l'instruction SCREEN « oublie » d'effacer l'écran.

3. Quelle couleur de fond obtient-on avec l'instruction COLOR 12 ?

☐ Le rouge saturé à 100 %
☐ Le rouge en demi-teinte

4. De combien de couleurs d'encre différentes dispose-t-on avec une seule palette ?

☐ 4
☐ 8
☐ 16

5. Avec quelle instruction allume-t-on un point dans une couleur d'encre ?

☐ PSET
☐ PRESET

6. Que dessine l'instruction LINE – (160, 100) ?

☐ Un trait horizontal
☐ Un trait vertical
☐ Un trait oblique

7. A quoi sert l'instruction STEP ?

☐ A indiquer au Basic que les coordonnées graphiques sont exprimées en valeurs absolues.
☐ A déterminer le pas d'incrémentation d'une boucle

(Réponses page 58)

CHAPITRE II

LES FIGURES GÉOMÉTRIQUES

La maîtrise de l'art graphique sur l'IBM PC ou ses compatibles passe nécessairement par la connaissance des instructions et des méthodes permettant d'obtenir les figures géométriques de base les plus employées. Si tel n'est pas votre cas, un rapide coup d'œil à ce chapitre peut s'avérer fructueux pour la suite à venir.

Menu du chapitre 2

Rectangles et boîtes
Cercles
Ellipses
Arcs de cercle et secteurs circulaires
Polygones
Etoiles

Quel est votre choix : ■

RECTANGLES ET BOÎTES

Créer un rectangle ou une boîte, ce n'est pas nouveau !... Au chapitre précédent figurait un exemple de dessin d'un cadre autour de l'écran destiné à illuster l'écriture simplifiée de l'instruction LINE. Pour aller un peu plus loin cette fois-ci, voici un second exemple destiné (lui aussi...) à dessiner un rectangle à partir des coordonnées de l'angle supérieur gauche et de l'angle inférieur droit :

Coordonnées de l'angle supérieur gauche... (→ ligne 30)

... et de l'angle inférieur droit ! (→ ligne 40)

```
10 KEY OFF
20 SCREEN 2:CLS
30 X1=0:Y1=0
40 X2=639:Y2=199
50 LINE (X1,Y1)-(X2,Y1),1
60 LINE-(X2,Y2),1
70 LINE-(X1,Y2),1
80 LINE-(X1,Y1),1
```

Dessin du rectangle (→ lignes 50 à 80)

LINE

OPTION B

Vous noterez que cet exemple nécessite 4 utilisations de l'instruction **LINE** (*) pour dessiner un rectangle. Or, c'est mal connaître cette instruction que de l'utiliser ainsi !... En effet, il sufit de rajouter la lettre B (pour boîte) à la suite de l'indication de la couleur (en intercalant une virgule...) pour aboutir au même résultat :

L'option B dessine des boîtes

```
10 KEY OFF
20 SCREEN 2:CLS
30 X1=0:Y1=0
40 X2=639:Y2=199
50 LINE (X1,Y1)-(X2,Y2),1,B
```

Ne pas oublier la virgule de séparation !

C'est tout de même plus élégant ainsi !... De surcroît, il est même possible de dessiner cette boîte rectangulaire en traits pointillés en rajoutant à la suite de l'option B le code hexadécimal d'allumage et d'extinction des points lumineux. On aboutit alors à ceci :

Code d'allumage des pixels du rectangle

```
10 KEY OFF
20 SCREEN 2:CLS
30 X1=0:Y1=0
40 X2=639:Y2=199
50 LINE (X1,Y1)-(X2,Y2),1,B,&H3333
```

Quel progrès si on compare ce programme à son homologue du chapitre précédent. Non seulement, l'écriture en est plus rapide, mais l'exécution aussi !... C'est dire la puissance de l'instruction **LINE**.

Ceci dit, il est temps de quitter les rectangles pour d'autres figures toutes aussi intéressantes. Mais avant, voici un petit programme de démonstration destiné à illustrer la très grande rapidité d'exécution de l'option B de l'instruction **LINE** :

Cette fois-ci, c'est le hasard qui détermine les coordonnées

```
10 KEY OFF
20 SCREEN 2:CLS
30 X1=INT(RND*640):Y1=INT(RND*200)
40 X2=INT(RND*640):Y2=INT(RND*200)
50 LINE (X1,Y1)-(X2,Y2),1,B
60 GOTO 30
```

(*) La fiche détaillée de l'instruction LINE figure à la page 318 du volume 2 de Pratique des IBM-PC et compatibles.

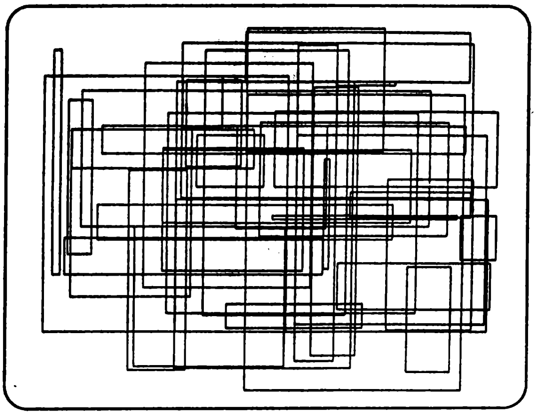


Fig. 2.1. : Des rectangles à ne plus savoir où les mettre.

> **L'option B de l'instruction LINE dessine un rectangle caractérisé par les coordonnées de ses angles supérieur gauche et inférieur droit.**

CERCLES

Le cercle est la figure géométrique qui, par ordre d'importante, arrive en deuxième position au « hit-parade » des figures. C'est dire son importance !

Comment dessine-t-on un cercle ?... Mais en se servant de l'une des trois relations connues (par l'auteur, s'entend...), à savoir la relation trigonométrique, pythagorienne ou celle des séries alternées. Seule la relation trigonométrique retiendra notre attention, car c'est elle qui offre le plus de facilités pour dessiner des secteurs circulaires (mais n'anticipons pas...)

Fort de ce choix, voici un petit programme qui, après avoir défini les coordonnées X0, Y0, du centre du cercle et le rayon R de celui-ci, se charge d'en dessiner point par point la circonférence en calculant pour chacun d'eux la projection sur chacun des axes du vecteur rayon tournant de 0 à 2π radians (ouf !) :

Coordonnées du centre et rayon du cercle

Calcul des projections du rayon R sur l'axe des X et des Y

```
10 KEY OFF
20 SCREEN 2:CLS
30 PI=3.141593
40 XO=320:YO=100
50 RAYON=200
60 FOR ANGLE=0 TO 2*PI STEP 1/RAYON
70 X=XO+RAYON*COS(ANGLE)
80 Y=YO+RAYON*SIN(ANGLE)
90 PSET(X,Y)
100 NEXT
```

Dessin du cercle

En fait de cercle, c'est une immense ellipse que l'on obtient dont une grande partie ne tient pas dans l'écran. Déception ?... Non, réflexion !... En y regardant à deux fois, ce qui se passe est tout à fait normal puisque la résolution verticale de l'écran graphique à haute résolution n'est que la moitié de sa résolution horizontale. Dans ces conditions, il suffit d'en tenir compte dans la ligne 70 en divisant le résultat obtenu par un facteur 2 :

Ce facteur 2 devrait rendre le cercle rond

```
70 Y=YO+RAYON*SIN(ANGLE)/2
```

Le résultat se fait attendre un peu (c'est vrai que l'exécution du dessin n'est pas rapide...) et voici qu'apparaît... encore une ellipse !... Décidément, il est plus facile d'obtenir une ellipse qu'un cercle parfait !... Néanmoins, celle-ci est un peu moins « elliptique » que la précédente puisque, à quelques pixels près, elle tient entièrement dans l'écran.

A quoi cela tient-il cette fois-ci ?... Tout simplement à la taille du pixel élémentaire !... Il faut que vous sachiez que les pixels de votre écran graphique IBM ne sont ni ronds, ni carrés, mais qu'ils s'inscrivent dans un rectangle plus haut que large. Si maintenant l'on vous dit que le rapport largeur/hauteur est égal à 5/6, vous en déduisez immédiatement que la valeur du facteur de correction à employer en mode graphique à haute résolution doit être de 5/12 (5/6 . 1/2) et non 1/2 comme précédemment.

Ceci dit, il vous suffit de reporter cette valeur de 5/12 dans la ligne 70 pour voir enfin apparaître un beau cercle bien rond :

Facteur de correction à employer en mode graphique à haute résolution

```
70 Y=YO+RAYON*SIN(ANGLE)*5/12
```

Remarque : *Le facteur de correction à employer en mode graphique à moyenne résolution est égal à 5/6, c'est-à-dire le double de celui du mode graphique à haute résolution.*

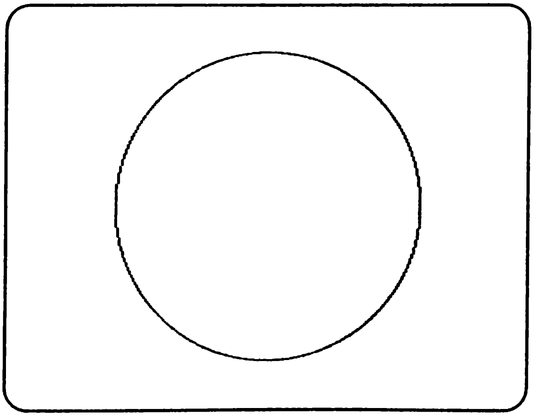


Fig. 2.2. : Cercle qui doit être parfaitement rond si l'amplitude verticale de votre moniteur IBM est bien réglée.

Il est bon de noter que c'est l'occasion ou jamais pour régler l'aptitude verticale de votre moniteur IBM. Bien réglé, ce dernier doit vous afficher un cercle parfaitement rond dénué de toute « ellipsoïdalité ».

CIRCLE

Ceci étant, que diriez-vous si l'on vous apprenait qu'il existe dans le Basic IBM une instruction dénommée **CIRCLE** (*) capable à elle toute seule de remplacer les lignes 50 à 90 ?... Génial non !... Côté syntaxe d'écriture, cette instruction **CIRCLE** nécessite que lui soient indiquées (entre parenthèses) les coordonnées du centre du cercle, suivies d'une virgule et de la valeur du rayon. Quant au facteur de correction, on n'en parle pas. Il est égal par défaut à 5/6 pour le mode graphique à moyenne résolution et à 5/12 pour le mode graphique à haute résolution.

C'est ainsi que pour reprendre les valeurs du programme précédent, il vous suffit d'écrire :

```
10 KEY OFF
20 SCREEN 2:CLS
30 XO=320:YO=100
40 RAYON=200
50 CIRCLE(XO,YO),RAYON
```

... et rayon du cercle

Coordonnée du centre...

... pour obtenir un cercle aussi beau que le précédent, mais dessiné ô combien plus rapidement !

Remarque : *Au sujet de l'instruction CIRCLE, il est bon de savoir que lorsque le rayon est très grand, les points en dehors de l'écran ne sont pas dessinés. De même, rien n'interdit au centre d'être positionné en dehors des limites de l'écran.*

Le tracé de cercles s'opère très rapidement à l'aide de l'instruction CIRCLE.

ELLIPSES

Comme vous avez pu le découvrir lors du tracé d'un cercle, il est plus facile de dessiner une ellipse qu'un cercle parfaitement rond. Est-ce à dire que dessiner une ellipse est une opération toute simple qui se passe de commentaires ?.. Rien n'est moins évident !

A titre de démonstration de ce qui vient d'être dit, vous allez tenter de dessiner une ellipse horizontale dont le rapport des grands axes soit égal à 2, ce qui revient à dire que l'axe horizontal doit être deux fois plus grand que l'axe vertical.

Comme vous pouvez vous en douter, il va vous falloir intervenir sur la valeur du facteur de correction. La question est de savoir où il faut placer ce dernier à la suite de l'instruction **CIRCLE** ?... Mais à la suite de l'indication du rayon, en prenant la précaution d'intercaler 4 virgules (c'est dire le nombre d'options de cette instruction...). C'est ainsi que dans une première approche vous pouvez essayer ce qui suit :

(*) La fiche détaillées de l'instruction CIRCLE figure à la page 304 du volume 2 de Pratique des IBM-PC et compatibles.

L'indication du facteur de correction...

```
10 KEY OFF
20 SCREEN 2:CLS
30 CIRCLE (320,100),200,,,,2
```

... à la suite de 4 virgules !

Le moins que l'on puisse dire, c'est que le résultat obtenu ne répond pas exactement à ce que l'on attendait !... Le grand axe de l'ellipse est vertical alors qu'on le souhaitait horizontal. Qu'à cela ne tienne, il suffit d'inverser le facteur de correction à la fin de la ligne 30, ce qui donne 1/2 au lieu de 2 :

```
30 CIRCLE (320,100),200,,,,1/2
```

Inversion du facteur de correction

Le résultat est-il meilleur ?... Non, pas précisément !... Cette fois-ci, l'ellipse est un peu moins dilatée vers le haut, mais on ne peut pas dire qu'elle correspond à ce qu'on attendait d'elle. Alors, où est le problème ?

Le problème réside dans le fait qu'il faut déjà appliquer un facteur de correction de 5/12 (5/6 en moyenne résolution) lorsque nous souhaitons dessiner un cercle qui, somme toute, n'est rien d'autre qu'une ellipse dont le rapport des grands axes est égal à 1. Dans ces conditions, il faut tenir compte de cette pré-correction dans le calcul du facteur de correction de l'ellipse. Ainsi, dans l'exemple qui est le nôtre, il faut que celui-ci soit égal à 5/24 (5/12 . 1/2) :

```
30 CIRCLE (320,100),200,,,,5/24
```

Facteur de correction égal au rapport des grands axes multiplié par 5/12

... ce qui confirme aussitôt l'essai :

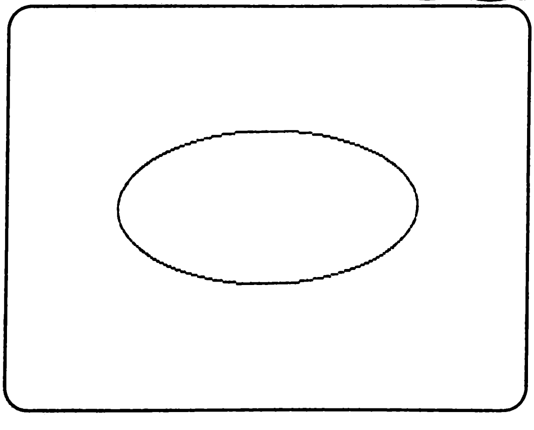


Fig. 2.3. : Ellipse dont le rapport des grands axes est égal à 1/2 (facteur de correction = 5/24)

En conclusion, il est important de noter que dans le dessin d'une ellipse, le rapport des grands axes correspond au rapport de l'axe vertical sur l'axe horizontal et que celui-ci doit être multiplié par 5/12 (ou 5/6 en moyenne résolution) avant d'être introduit comme facteur de correction dans l'instruction CIRCLE.

Pour terminer en beauté cette partie sur les ellipses, voici un petit programme qui est censé (sait-on jamais...) dessiner la planisphère avec ses méridiens :

```
10 KEY OFF
20 SCREEN 2:CLS
30 PI=3.141593
40 FOR MERIDIEN=0 TO 6
50 ANGLE=MERIDIEN*PI/12
60 RAYON=200*COS(ANGLE):IF MERIDIEN>4 THEN RAYON=82.6
70 FACTEUR=5/12/COS(ANGLE):IF MERIDIEN=6 THEN FACTEUR=513
80 CIRCLE (320,100),RAYON,,,,FACTEUR
90 NEXT
```

A quoi peuvent bien correspondre ces valeurs ?

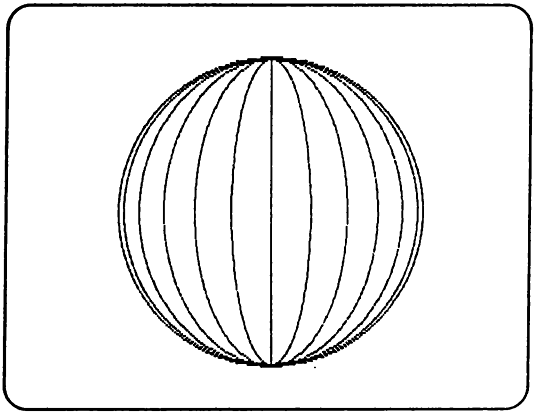


Fig. 2.4. : La planisfère avec ses méridiens dessinés à partir d'ellipses.

Normalement, si vous êtes curieux, vous devirez être intrigué par les valeurs numériques 82,6 et 513 !... Pour la première d'entre-elles, il vous faut savoir que l'instruction **CIRCLE** dessine des ellipses horizontales à grand axe constant tant que le facteur de correction reste inférieur à 1 et des ellipses verticales à grand axe constant dans le cas où il est supérieur à 1. Si vous faites le calcul pour les différents méridiens, vous découvrirez que ce facteur est inférieur à 1 pour les méridiens 0 à 4 et supérieur à 1 pour les méridiens 5 et 6. Il a donc fallu compenser la « commutation » de mode de fonctionnement de l'instruction **CIRCLE** à partir du 5e méridien, en attribuant une nouvelle valeur au rayon (82,6 en l'occurence !).

Quant à la valeur 513, c'est celle du facteur de correction à partir de laquelle vous n'obtenez qu'un trait vertical en guise d'ellipse.

Le dessin d'une ellipse s'obtient en attribuant au facteur de correction de l'instruction CIRCLE une valeur correspondant au rapport de l'axe vertical sur l'axe horizontal multiplié par 5/12 (5/6 en moyenne résolution).

ARCS DE CERCLE ET SECTEURS CIRCULAIRES

Un arc de cercle est une figure bien intéressante à savoir dessiner car c'est d'elle que va dépendre la (prochaine...) réalisation de nos graphiques de gestion en forme de camemberts. C'est pourquoi il y a lieu de se pencher quelque peu sui le mode opératoire qui permet d'aboutir à une telle figure.

Tout d'abord, qu'est-ce qu'un arc de cercle... C'est un cercle qui n'est pas terminé !... En d'autres termes, c'est un morceau de la circonférence d'un cercle dont on précise l'angle au centre de début et de fin. Et où doit-on placer ces valeurs ?... Mais très précisément à la suite de l'instruction **CIRCLE**, 2 virgules aprés l'indication du rayon et en débutant, comme de bien entendu, par l'indication de l'angle de début.

Pour illustrer ce qui précède, voici un exemple de tracé d'un arc compris entre π/4 et π radians :

Valeurs de début et de fin de l'arc de cercle...

```
10 KEY OFF
20 SCREEN 2:CLS
30 PI=3.141593
40 DEBUT=PI/4:FIN=PI
50 CIRCLE (320,100),200,,DEBUT,FIN,5/12
```

... que l'on place ici !

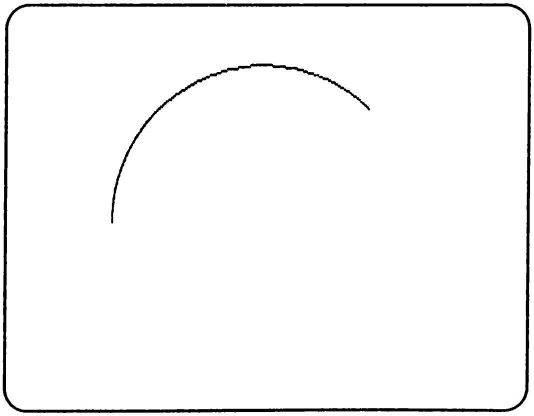
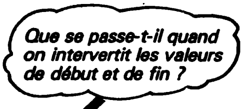


Fig. 2.5. : Dessin d'un arc de cercle entre π/4 et π radians.

Remarquez que l'arc de cercle est dessiné dans le sens de parcours du cercle trigonométrique, à savoir dans le sens inverse des aiguilles d'une montre.

En est-il toujours de même ?... Que se passe-t-il, par exemple, quand vous demandez à ce que l'arc de cercle soit dessiné entre π et π/4 ?... Un rapide essai va nous fournir instantanément la réponse :




```
10 KEY OFF
20 SCREEN 2:CLS
30 PI=3.141593
40 DEBUT=PI:FIN=PI/4
50 CIRCLE (320,100),200,,DEBUT,FIN,5/12
```

On obtient l'arc de cercle complémentaire au précédent, comme cela est visible sur la figure ci-après :

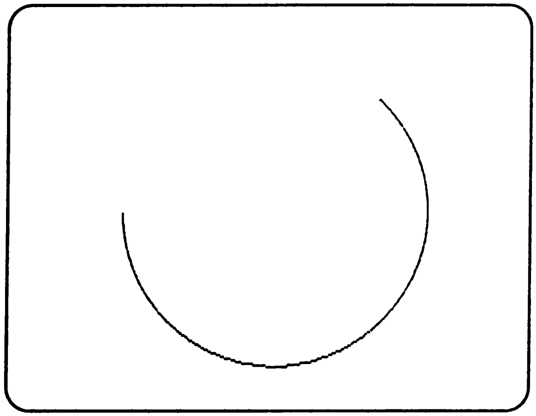


Fig. 2.6. : En intervertissant les valeurs de début et de fin, on obtient l'arc de cercle complémentaire.

De ceci, il est possible d'en tirer une règle : que la valeur de fin soit ou non plus grande que la valeur de début, l'instruction **CIRCLE** dessine toujours l'arc de cercle en tournant dans le sens trigonométrique.

Remarque : *Les valeurs de début et de fin d'un arc de cercle doivent être exprimées dans l'intervalle 0 - 2π. En dehors de ces valeurs, vous n'obtenez qu'un : ILLEGAL FUNCTION CALL.*

Voyons un peu maintenant ce qu'il en est des secteurs circulaires, c'est-à-dire des arcs de cercle délimités par deux rayons. Y a-t-il lieu pour dessiner les rayons de devoir calculer les coordonnées de début et de fin de chaque arc de cercle ?... Non, car l'instruction **CIRCLE** possède une option qui réalise cela directement à notre place. Il suffit de placer le signe moins devant l'une ou l'autre des deux valeurs pour que soit tracé le rayon correspondant, ce que confirme l'essai suivant :

Le signe moins dessine un rayon à l'extrémité de l'arc de cercle

```
50 CIRCLE (320,100),200,,-DEBUT,-FIN,5/12
```

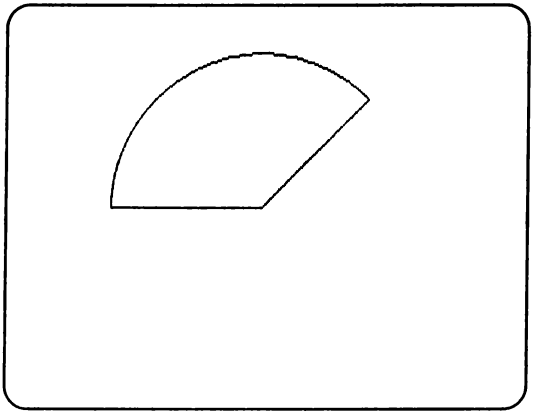


Fig. 2.7. : Secteur circulaire obtenu en plaçant un signe moins devant les valeurs de début et de fin de l'arc de cercle.

Ceci étant, voyons maintenant comment « extraire » un secteur circulaire d'un cercle de façon à le mettre en relief, à la manière d'une portion découpée dans un camembert. La méthode est relativement simple !... Il suffit après avoir déterminé le rayon de décalage de la portion, de calculer les projections DX et DY de la bissectrice de la portion que l'on veut mettre en relief. C'est ainsi que l'on aboutit au programme suivant :

Notez ici l'inversion des valeurs de début et de fin...

... de façon à dessiner le secteur complémentaire !

```
10 KEY OFF
20 SCREEN 2:CLS
30 PI=3.141593
40 XO=320:YO=100
50 RAYON=200
60 DEBUT=0:FIN=PI/4
70 CIRCLE (XO,YO),RAYON,,-FIN,-DEBUT,5/12
80 DECALAGE=RAYON/4
90 BISSECTRICE=(FIN-DEBUT)/2
100 DX=DECALAGE*COS(BISSECTRICE)
110 DY=DECALAGE*SIN(BISSECTRICE)*5/12
120 CIRCLE (XO+DX,YO-DY),RAYON,,-DEBUT,-FIN,5/12
```

Quant à cette ligne-ci, elle dessine la portion «éclatée»

... dans lequel la ligne 80 calcule la valeur du rayon de décalage, puis c'est au tour de la bissectrice du secteur à la ligne 90, suivie par le calcul des projections DX et DY aux lignes 100 et 110. Notez qu'en ce qui concerne la projection DY, il a fallu tenir compte du facteur de correction habituel de 5/12.

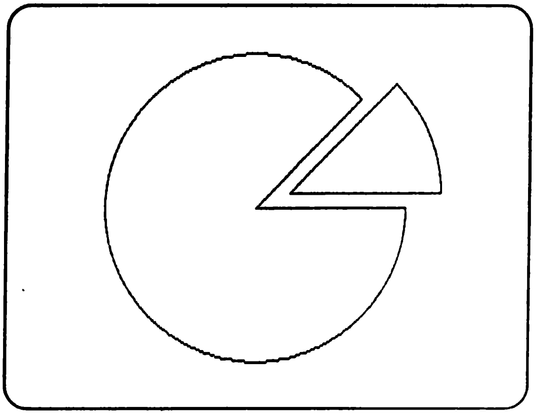


Fig. 2.8. : Secteur circulaire extrait d'un cercle.

> **Le dessin d'arcs de cercle s'obtient en précisant dans l'instruction CIRCLE les valeurs de début et de fin de l'arc. Quant à obtenir un secteur circulaire, il suffit de faire précéder ces valeurs d'un signe moins.**

POLYGONES

Du cercle au polygone il n'y a qu'un pas à franchir car qu'est-ce qu'un cercle si ce n'est un polygone à très grand nombre de côtés... Toutefois, l'analogie s'arrête là, l'instruction CIRCLE étant totalement inopérante pour dessiner un polygone.

Dans ces conditions, comment faire pour obtenir un polygone ?... Il suffit de déterminer sur une circonférence fictive de cercle (que l'on ne trace pas) autant de points équirépartis que vous souhaitez de côtés au polygone et de relier entre eux par un trait les points adjacents deux à deux. D'ailleurs, plutôt que de longues explications, essayez plutôt ce petit programme :

```
10 KEY OFF
20 SCREEN 2:CLS
30 PI=3.141593
40 XO=320:YO=100
50 RAYON=200
60 COTES=5
70 PAS=.0001*INT(10000*2*PI/COTES)
80 FOR ANGLE=0 TO 2*PI STEP PAS
90 X=XO+RAYON*COS(ANGLE)
100 Y=YO-RAYON*SIN(ANGLE)*5/12
110 IF ANGLE=0 THEN PRESET(X,Y)
120 LINE -(X,Y)
130 NEXT
```

Nombre de côtés du polygone (ligne 60)

Calcul du pas arrondi à 4 décimales (ligne 70)

Positionnement du point de départ (ligne 110)

Dessin des côtés du polygone (ligne 120)

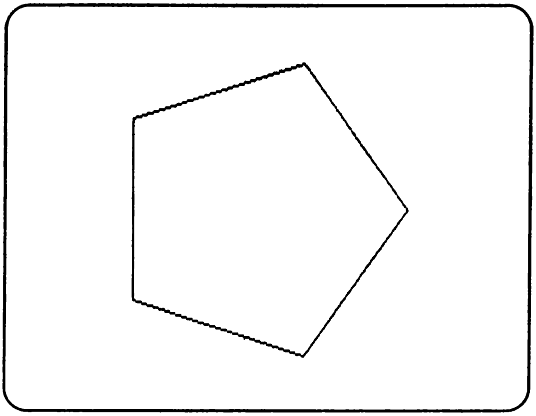


Fig. 2.9. : Pentagone (polygone à 5 côtés)

Question à cent sous : Pourquoi avoir arrondi à 4 décimales le calcul du pas de la boucle FOR... TO ?... Parce que, sans cela, les règles d'arrondi automatique de la machine font que, pour certains polygones (l'hexagone en est un exemple), le dernier côté ne peut être tracé car la variable de boucle (ANGLE en l'occurence) dépasse d'un quart de tiers de « micropoil » la valeur de fin de boucle. Ainsi, en établissant un arrondi obligatoire par défaut, on s'affranchit de cet inconvénient.

Pour le reste, le programme est sans surprise !... A chaque parcours de la boucle qui, soit dit en passant, détermine autant de points équidistants sur la circonférence d'un cercle fictif qu'il y a de côtés au polygone (ligne 80), le programme calcule les coordonnées de ces points (lignes 90 et 100) et dessine un trait entre les coordonnées venant d'être calculées et l'ancien point mis en mémoire (ligne 120), non sans avoir pensé à fixer au préalable les coordonnées du point de départ (ligne 110).

Comme cela, vous n'aimez pas l'allure du pentagone ?... Ah, vous lui trouvez un air de travers !... C'est normal !... Cela provient de la position du point de départ du dessin qui respecte les règles de la trigonométrie classique. Pour vous affranchir de ce second inconvénient, il vous faut introduire un angle de décalage dans la boucle **FOR... TO** de façon à placer le point de départ où bon vous semble.

Si l'on reprend l'exemple précédent, l'introduction de ce décalage dans le programme peut se faire ainsi :

```
10 KEY OFF
20 SCREEN 2:CLS
30 PI=3.141593
40 X0=320:Y0=100
50 RAYON=200
60 COTES=5
70 DECALAGE=PI/2
80 PAS=.0001*INT(10000*2*PI/COTES)
90 FOR ANGLE=DECALAGE TO 2*PI+DECALAGE STEP PAS
100 X=X0+RAYON*COS(ANGLE)
110 Y=Y0-RAYON*SIN(ANGLE)*5/12
120 IF ANGLE=DECALAGE THEN PRESET(X,Y)
130 LINE -(X,Y)
140 NEXT
```

Valeur de l'angle de décalage (ligne 70)

Introduction du décalage dans la boucle... (ligne 90)

... ainsi que dans la fixation du point de départ ! (ligne 120)

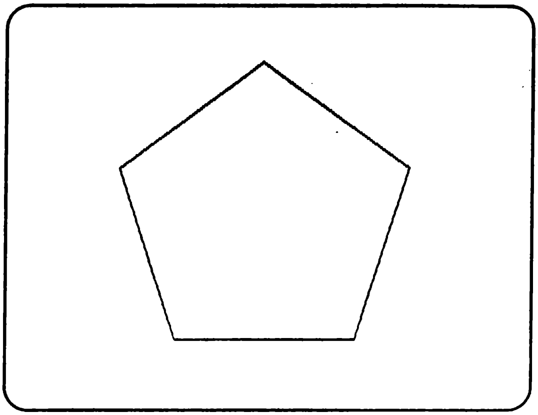


Fig. 2.10. : Comparé au précédent, ce pentagone a subi un décalage angulaire de π/2.

Vous noterez que pour rétablir l'allure du pentagone, c'est-à-dire pour rendre horizontal l'un de ses côtés, il a suffit de lui faire subir un décalage de π/2 au départ (ligne 90), décalage que l'on retrouve forcément à la ligne 120 de fixation du point de départ.

Poùr dessiner un polygone, il suffit de déterminer sur une circonférence fictive autant de points équirépartis que le polygone comporte de côtés et de relier entre eux deux à deux les points adjacents.

ÉTOILES

Au tour des étoiles maintenant !... Savez-vous dessiner une étoile ?... Non ?... Alors, suivez le guide !... Pour obtenir une étoile, il faut tout d'abord placer dans un tableau les coordonnées des extrémités des branches. Ce faisant, il suffit alors de dessiner un trait entre un point et le point qui lui est le plus éloigné (à l'exception du point diamétralement opposé), de façon à tracer les traits les plus longs qu'il soit. L'étoile est terminée (enfin presque...) lorsque le dernier trait rejoint le point de départ.

Voyons un peu tout cela sur le petit programme ci-après. Vous reconnaîtrez au passage le dimensionnement du tableau des coordonnées X, Y des extrémités des branches qui dépend (forcément...) du nombre de branches souhaitées. Dans le cas de cet exemple, c'est une étoile à 5 branches que vous devriez obtenir :

Mise en tableau des coordonnées des extrémités des branches de l'étoile (lignes 80 à 120)

Dessin des traits de l'étoile (ligne 180)

```
10 KEY OFF
20 SCREEN 2:CLS
30 BRANCHES=5
40 DIM X(BRANCHES),Y(BRANCHES)
50 PI=3.141593
60 X0=320:Y0=100
70 RAYON=200
80 FOR I=1 TO BRANCHES
90 ANGLE=PI/2+I*2*PI/BRANCHES
100 X(I)=X0+RAYON*COS(ANGLE)
110 Y(I)=Y0-RAYON*SIN(ANGLE)*5/12
120 NEXT
130 INCREMENT=INT((BRANCHES-1)/2)
140 I=1
150 PRESET (X(I),Y(I))
160 I=I+INCREMENT
170 IF I>BRANCHES THEN I=I-BRANCHES
180 LINE -(X(I),Y(I))
190 IF I<>1 THEN 160
```

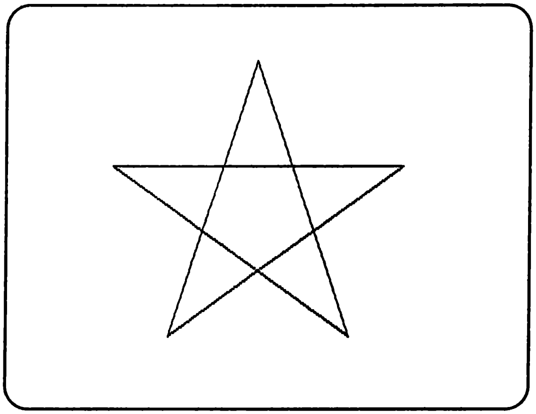


Fig. 2.11. : Etoile à 5 branches

Bien que simple d'apparence, ce programme nécessite quelques compléments d'information sur deux ou trois points délicats. Le premier d'entre-eux concerne la numérotation des branches de l'étoile, numérotation qui va de 1 à 5 en débutant par la branche située en haut de l'écran et en tournant dans le sens inverse des aiguilles d'une montre (bien que rien n'y oblige...). Le second point a trait au calcul de l'incrément (ligne 130) qui, rajouté au numéro d'ordre d'un point, doit permettre de trouver le point qui en est le plus éloigné. Vous noterez qu'il s'obtient en divisant par 2 le nombre de branches diminué d'une unité (ça, c'est pour éviter de tomber sur le point diamétralement opposé), résultat dont on ne prend que la partie entière (ça, c'est pour tenir compte des étoiles à nombre impair de branches).

Quant au dernier point (le plus délicat...), il concerne une certaine catégorie d'étoiles à 6, 10, 14, 18, etc... branches que le programme ne sait pas dessiner. Pour vous en convaincre, essayez donc de dessiner une étoile à 6 branches en modifiant la valeur de la ligne 30 !... Vous n'obtiendrez qu'un (beau) triangle la pointe en bas.

A quoi cela est-il dû ?... Mais au fait que ces étoiles comportent un nombre impair de paires de branches (3 × 2, 5 × 2, etc...) et que dans ces conditions on n'en dessine que la moitié en se rebouclant sur le point de départ. La solution ?... Elle consiste tout simplement à procéder à deux phases de dessin, l'une débutant par le point du haut et l'autre par le point du bas, comme le montre le programme suivant qui n'est rien d'autre que le programme précédent auquel on a adjoint quelques (légères...) modifications :

```
10 KEY OFF
20 SCREEN 2:CLS
30 BRANCHES=6
40 DIM X(BRANCHES),Y(BRANCHES)
50 PI=3.141593
60 X0=320:Y0=100
70 RAYON=200
80 FOR I=1 TO BRANCHES
90 ANGLE=PI/2+I*2*PI/BRANCHES
100 X(I)=X0+RAYON*COS(ANGLE)
110 Y(I)=Y0-RAYON*SIN(ANGLE)*5/12
120 NEXT
130 INCREMENT=INT((BRANCHES-1)/2)
140 DEBUT=1:TRAITS=0
150 I=DEBUT
160 PRESET (X(I),Y(I))
170 I=I+INCREMENT
180 IF I>BRANCHES THEN I=I-BRANCHES
190 LINE -(X(I),Y(I))
200 TRAITS=TRAITS+1
210 IF I<>DEBUT THEN 170
220 IF TRAITS=BRANCHES/2 THEN DEBUT=1+TRAITS:GOTO 150
```

Point de début (→ ligne 150)

Calcul du nombre de traits dessinés (→ ligne 200)

Si on n'obtient que la moitié des traits d'une étoile, alors on recommence en débutant cette fois-ci par le point du bas ! (→ ligne 220)

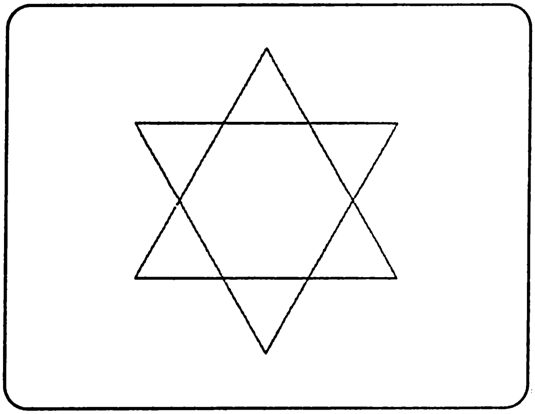


Fig. 2.12. : Etoile à 6 branches entièrement dessinée.

En ce qui concerne les modifications à introduire, vous noterez la présence de deux nouvelles variables, DEBUT (lignes 140, 150 et 200) et TRAITS (lignes 140, 190 et 210), qui représentent respectivement le numéro d'ordre du point de début de chaque phase de dessin et le nombre de traits dessinés. Ainsi, après avoir rebouclé le dessin sur le point n° 1, la ligne 210 vérifie si le nombre de traits dessinés est égal au nombre de branches de l'étoile. Si ce n'est pas le cas, elle attribue à la variable DEBUT le n° d'ordre du point du bas et recommence le dessin.

Pour dessiner une étoile, il suffit, après avoir placé dans un tableau les coordonnées des extrémités des branches, de relier chaque point avec celui qui lui est le plus éloigné, à l'exception du point diamétralement opposé.

Exercices sur le chapitre 2

1. Peut-on dessiner un rectangle à l'aide de l'instruction LINE ?

   ☐ Oui, à condition d'employer cette instruction 4 fois.
   ☐ Non, car c'est l'instruction BOX qui sert à cela.

2. Quelle valeur du facteur de correction faut-il rajouter à l'instruction CIRCLE pour obtenir un cercle parfaitement rond ?

   ☐ 5/6 ou 5/12 selon le mode graphique.
   ☐ C'est inutile car celle-ci est fixée par défaut.

3. A quoi correspond le facteur de correction de l'instruction CIRCLE lorsque l'on cherche à dessiner une ellipse ?

   ☐ Au rapport de l'axe vertical sur l'axe horizontal de l'ellipse
   ☐ Au rapport de l'axe horizontal sur l'axe vertical de l'ellipse

4. Comment dessine-t-on un secteur circulaire ?

   ☐ En employant des valeurs négatives de début et de fin d'arc.
   ☐ En intervertissant les valeurs de début et de fin d'arc.

5. Peut-on employer l'instruction CIRCLE pour dessiner un polygone ?

   ☐ Oui, à condition de préciser les valeurs d'angle de début et de fin de chaque côté du polygone
   ☐ Non, elle ne le peut pas

6. Dans quel cas faut-il procéder au dessin d'une étoile en deux passes ?

   ☐ Lorsque le nombre de branche de l'étoile est pair
   ☐ Lorsque le nombre de branches de l'étoile est impair.

(Réponses page 77)

CHAPITRE III

COLORIAGE DES GRAPHIQUES

Après avoir dessiné le contour des figures géométriques, vous souhaitez peut-être les remplir de couleurs, ne serait-ce que pour mieux les distinguer les unes des autres ?... Cela va vous être montré dans le présent chapitre, ainsi que l'art et la matière d'obtenir en mode graphique à haute résolution diverses nuances de gris destinées, elles aussi, a distinguer les figures les unes des autres dans un univers limité au noir et blanc.

Menu du chapitre 3

Coloriage avec l'option F
Coloriage avec l'instruction PAINT
Des couleurs supplémentaires avec LINE
Des couleurs en supplément avec PAINT
Les couleurs en noir et blanc
Coloriage en haute résolution

Quel est votre choix : ■

COLORIAGE AVEC L'OPTION F

Remplir une figure avec une même couleur est une opération qui porte le nom de coloriage et qui peut, selon le cas, nécessiter une fraction de seconde comme plusieurs secondes de temps d'exécution.

Pourquoi une telle différence ?... Cela tient à ce que certaines méthodes de coloriage s'obtiennent à l'aide de programmes écrits en Basic, alors que d'autres font directement appel au langage machine via le biais d'une instruction Basic.

Pour mieux cerner ce point important (de lui dépend toute la qualité d'une animation vidéo), voici un petit exercice qui a pour but de remplir l'intérieur d'un rectangle avec une même couleur :

Dessin du pourtour du rectangle en jaune...

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 8,0
40 COULEUR=2
50 POURTOUR=3
60 X1=80:Y1=50
70 X2=240:Y2=150
80 LINE(X1,Y1)-(X2,Y2),POURTOUR,B
90 FOR Y=Y1+1 TO Y2-1
100 LINE (X1+1,Y)-(X2-1,Y),COULEUR
110 NEXT
```

...et son remplissage avec la couleur rouge !

Vous noterez que le remplissage, loin d'être instantané, nécessite de 2 à 3 secondes. C'est trop, beaucoup trop !...

LINE

OPTION F

Heureusement, il existe une autre solution plus rapide qui consiste à employer l'option F (de l'anglais FILL pour remplissage) de l'instruction **LINE**. Placée à la suite immédiate de l'option B (sans virgule de séparation), elle va remplir quasi-instantanément le rectangle avec, cette fois-ci la même couleur que celle du pourtour :

L'option F remplit le rectangle avec la même couleur que le pourtour

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 8,0
40 POURTOUR=3
50 X1=80:Y1=50
60 X2=240:Y2=150
70 LINE(X1,Y1)-(X2,Y2),POURTOUR,BF
```

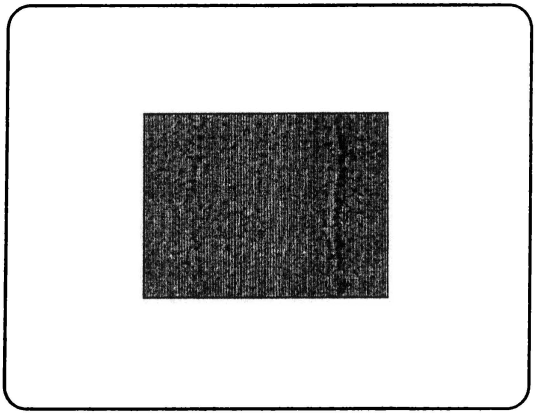


Fig. 3.1. : L'option F de l'instruction LINE remplit un rectangle avec la même couleur que celle du pourtour.

L'inconvénient avec cette option, c'est qu'on perd la possibilité de dessiner le pourtour du rectangle avec une couleur différente de celle de l'intérieur. Toutefois, est-ce bien vrai ?... Essayez donc ce petit programme pour voir :

Dessin du pourtour du rectangle en jaune... (ligne 80)

... et son remplissage ultra-rapide avec la couleur rouge ! (ligne 90)

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 8,0
40 COULEUR=2
50 POURTOUR=3
60 X1=80:Y1=50
70 X2=240:Y2=150
80 LINE(X1,Y1)-(X2,Y2),POURTOUR,B
90 LINE(X1+1,Y1+1)-(X2-1,Y2-1),COULEUR,BF
```

Comme vous pouvez le constater, vous obtenez à nouveau le pourtour du rectangle dans une couleur et l'intérièur dans une autre.

Remarque : *L'emploi de l'option F interdit l'utilisation du code d'allumage des points. C'est l'un ou l'autre !*

L'option F de l'utilisation LINE assure le remplissage ultra-rapide d'un rectangle.

COLORIAGE AVEC L'INSTRUCTION PAINT

Tout ceci est bien beau me direz-vous, mais l'option F de l'instruction LINE ne permet de colorier que l'intérieur des rectangles. Comment faire pour colorier les autres figures ?...

PAINT

Mais en employant tout simplement l'instruction PAINT (*) qui, comme son nom le laisse supposer, sert à « peindre » l'intérieur des figures qu'on lui soumet. Est-ce à dire que cette instruction PAINT est capable de remplir n'importe quelle figure ?... Seule l'expérience nous le dira !

Voici donc pour commencer un petit exercice de coloriage de l'intérieur d'un cercle. Vous noterez au passage qu'il faut indiquer à l'instruction PAINT les coordonnées du point à partir duquel le coloriage doit débuter et que, dans le cas d'un cercle, le plus simple (et le plus sûr aussi...) consiste à choisir les coordonnées du centre :

Dessin d'un cercle... (ligne 40)

... suivi de son coloriage intérieur avec PAINT ! (ligne 50)

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 8,0
40 CIRCLE (160,100),200
50 PAINT(160,100)
```

(*) La fiche détaillée de l'instruction LINE figure à la page 318 du volume 2 de Pratique des IBM-PC et compatibles.

Comme vous pouvez le constater, vous obtenez un beau disque jaune. Pourquoi jaune ?... Parce que c'est la couleur sélectionnée par défaut (couleur n° 3, palette n° 0) par les instructions CIRCLE et PAINT lorsqu'on omet l'indication de couleur.

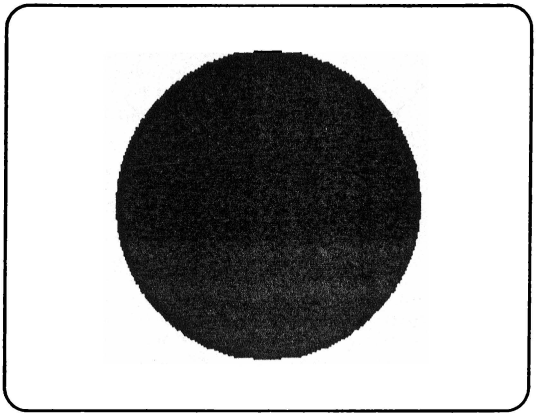
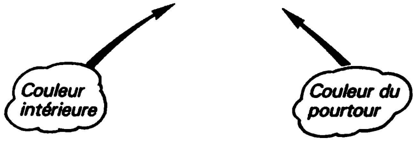


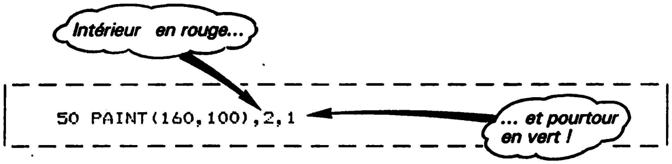
Fig. 3.2. : Disque plein obtenu avec l'instruction PAINT.

Et si l'on veut une autre couleur que le jaune ?... Dans ce cas, il suffit de le dire (ou plutôt de l'écrire...) à l'instruction **PAINT**. Sa syntaxe d'écriture étant :

PAINT (colonne, ligne), intérieur, pourtour.



... il vous suffit de modifier comme suit la ligne 50 pour obtenir un beau disque rouge entouré de vert :



Tiens, cela ne marche pas !... Tout ce qu'on obtient, c'est un écran entièrement rouge comme si l'instruction **PAINT** ne reconnaissait pas la circonférence du cercle présent sur l'écran. A quoi cela peut-il être dû ? Mais tout simplement au fait que l'instruction **PAINT** ne sait reconnaître un contour qu'à la condition expresse qu'on lui précise une couleur de contour qui soit la même que celle avec laquelle le contour en question a été dessiné. Sinon, comment voulez-vous qu'elle s'y retrouve !

Ainsi, dans notre exemple, si l'on veut peindre l'intérieur du cercle en rouge et le pourtour en vert, il faut que le cercle ait été préalablement dessiné en vert, et non en jaune comme c'est le cas, ce que confirme aussitôt l'essai suivant :

Si l'on veut que PAINT fonctionne, il faut que la couleur du pourtour du cercle...

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 8,0
40 CIRCLE (160,100),100,1
50 PAINT(160,100),2,1
```

... soit la même que celle indiquée à la suite de PAINT

Remarque : *Dans le cas où l'on ne précise pas dans* **PAINT** *la couleur du pourtour, c'est cette fois-ci la couleur intérieure qui doit être la même que celle du cercle.*

Et maintenant, que se passe-t-il si on débute le coloriage par un point situé en dehors de la figure à colorier ?... Dans ce cas, c'est tout l'extérieur de la figure qui est colorié et non l'intérieur, comme le montre l'exemple que voici :

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 8,0
40 CIRCLE(160,100),100,2
50 PAINT(0,0),1,2
```

Pour colorier l'extérieur d'une figure...

... il suffit de débuter le coloriage par un point situé à l'extérieur !

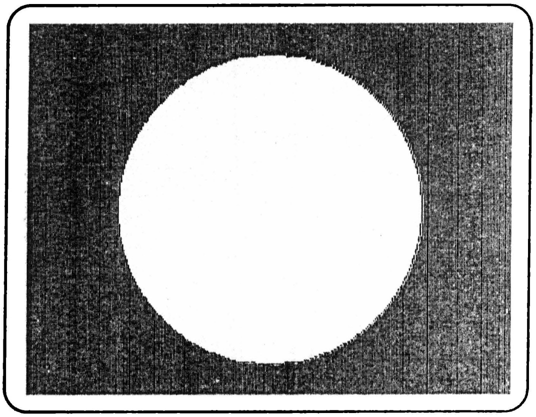


Fig. 3.3. : Coloriage de l'extérieur d'une figure.

Comme vous pouvez le constater, l'instruction PAINT est une instruction très puissante du Basic IBM (et vous n'avez pas tout vu...) qui vous rendra de nombreux services lorsqu'il s'agira de colorier les barres ou les portions de camembert de vos futurs graphiques de gestion.

Ah, encore une question !... L'instruction PAINT fonctionne-t-elle en mode graphique à haute résolution ?... Pour le savoir, il n'y a qu'à essayer avec le petit programme que voici qui n'est rien d'autre que celui avec lequel vous aviez dessiné une étoile au chapitre précédent :

```
10 KEY OFF
20 SCREEN 2:CLS
30 BRANCHES=5
40 DIM X(BRANCHES),Y(BRANCHES)
50 PI=3.141593
60 XO=320:YO=100
70 RAYON=200
80 FOR I=1 TO BRANCHES
90 ANGLE=PI/2+I*2*PI/BRANCHES
100 X(I)=XO+RAYON*COS(ANGLE)
110 Y(I)=YO-RAYON*SIN(ANGLE)*5/12
120 NEXT
130 INCREMENT=INT((BRANCHES-1)/2)
140 DEBUT=1:TRAITS=0
150 I=DEBUT
160 PRESET (X(I),Y(I))
170 I=I+INCREMENT
180 IF I>BRANCHES THEN I=I-BRANCHES
190 LINE -(X(I),Y(I))
200 TRAITS=TRAITS+1
210 IF I<>DEBUT THEN 170
220 IF TRAITS=BRANCHES/2 THEN DEBUT=1+TRAITS:GOTO 150
230 PAINT(320,100)
```

Programme de dessin d'une étoile en haute résolution...

... auquel on a greffé l'instruction PAINT !

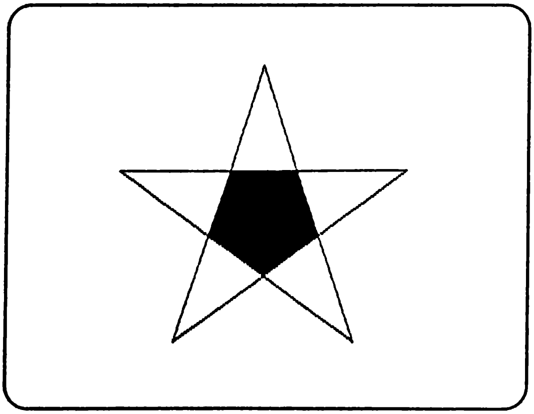
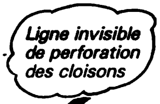


Fig. 3.4. : Seule la partie centrale de l'étoile est coloriée.

La réponse est oui, bien que seule la partie centrale de l'étoile ait été coloriée. Est-ce dépendant du mode graphique à haute résolution ?... Nullement !... Cela provient tout simplement du cloisonnement intérieur de l'étoile, cloisons que l'instruction **PAINT** reconnaît bien sagement comme les limites de la figure à colorier.

Est-ce sans solution ?... Non, bien sûr !... Une des nombreuses solutions consiste à perforer les cloisons de façon à laisser passer le coloriage vers chacune des extrémités des branches. Pour ce faire, il suffit de rajouter les quelques lignes suivantes qui dessinent un trait invisible (c'est-à-dire en noir) entre le milieu de chaque branche et le centre de l'étoile :




```
230 RAYON=RAYON/2
240 FOR I=1 TO BRANCHES
250 ANGLE=PI/2+I*2*PI/BRANCHES
260 X(I)=XO+RAYON*COS(ANGLE)
270 Y(I)=YO-RAYON*SIN(ANGLE)*5/12
280 LINE(X(I),Y(I))-(XO,YO),0
290 NEXT
```

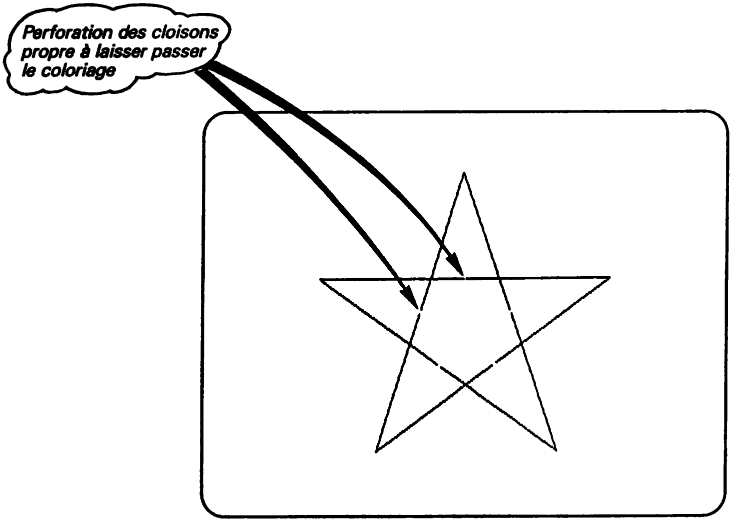


Fig. 3.5. : Perforation des cloisons de l'étoile.

Les cloisons étant perforées, il n'y a plus qu'à rajouter à la suite de la ligne 290 l'instruction **PAINT** de coloriage et le tour est joué !... La preuve :

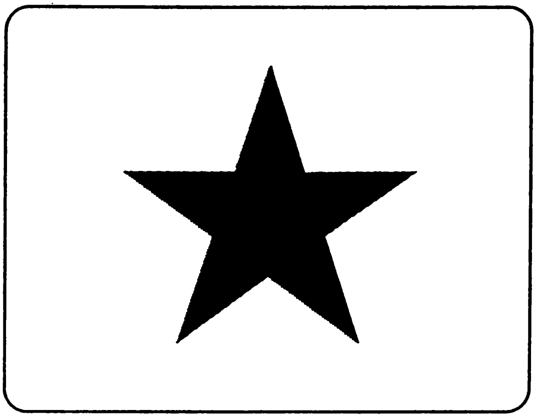


Fig. 3.6. Cette fois-ci, l'étoile est entièrement coloriée.

L'instruction PAINT permet de colorier l'intérieur ou l'extérieur des figures selon que le point de départ du coloriage est situé à l'intérieur ou à l'extérieur de la figure.

DES COULEURS SUPPLÉMENTAIRES AVEC LINE

Comme vous avez pu le constater par vous-même , l'IBM-PC que vous avez entre les mains ne peut vous proposer que 4 couleurs pour créer une image graphique. C'est peu !... Si au moins, on pouvait disposer simultanément des 8 couleurs des deux palettes, cela serait déjà mieux !

Eh bien, sachez que ce vœu pieux est réalisable à l'aide de l'instruction **LINE** et de son code d'allumage des points lumineux !...

Si l'on se penche quelques instants sur les notions de colorimétrie, on découvre qu'entre les couleurs de la palette n° 0 et celles de la palette n° 1, il n'y a que la couleur bleue de différence.

PALETTE N° 0	
Code	**Couleur**
0	Fond
1	Vert
2	Rouge
3	Jaune

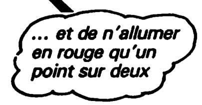
PALETTE N° 1		
Code	**Couleur**	**Mélange**
0	Fond	
1	Turquoise	Bleu + Vert
2	Violet	Bleu + Rouge
3	Blanc	Bleu + Jaune

En effet, le blanc étant obtenu par synthèse additive des trois couleurs primaires que sont le rouge, le vert et le bleu, on voit que la palette n° 0 n'emploie que 2 couleurs primaires, le vert et le rouge, tandis que la palette n° 1 les emploie toutes les trois avec une permanence du bleu. L'astuce, car c'en est une belle, consiste donc à employer les couleurs de la palette n° 1 et à les mélanger avec un fond bleu pour obtenir celles de la palette n° 1.

Et comment accomplit-on un tel mélange ?... Mais grâce au code d'allumage des points de l'instruction LINE !... Imaginez que vous souhaitez obtenir un trait violet. Il vous suffit pour cela de disposer d'un fond bleu et de tracer un trait rouge en n'allumant qu'un point sur deux (codes &H5555 ou &HAAAA). Vous allez obtenir ainsi une succession de points rouges et bleus, propres à créer l'illusion du violet.

Mais assez parlé, passons aux actes !... Voici un petit programme qui va se charger de colorier l'intérieur d'un rectangle en violet tout en ne disposant que des couleurs de la palette n° 0 :

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 9,0
40 FOR Y=50 TO 150 STEP 2
50 LINE(80,Y)-(240,Y),2,,&H5555
60 LINE(80,Y+1)-(240,Y+1),2,,&HAAAA
70 NEXT
```



Question à cent sous : Pourquoi utiliser 2 fois l'instruction **LINE** pour colorier en violet ?... C'est pour éviter de créer des alignements verticaux de points !... Si vous regardez de près les lignes 50 et 60, vou verrez qu'elles ne diffèrent (à l'exception des coordonnées en X, Y) que par leur code d'allumage, celui de la ligne 50 allumant un point sur deux en débutant par un point éteint (&H5555) tandis que celui de la ligne 60, qui n'allume également qu'un point sur deux, débute par un point allumé. Vous obtenez ainsi un quadrillage « en quinconce » moins discernable à l'œil.

Puisque cela fonctionne, amusez-vous donc à obtenir les deux autres couleurs que sont le turquoise et le blanc en mélangeant entre-eux du vert et du bleu (pour le turquoise) ou du jaune et du bleu (pour le blanc).

Mieux encore !... Vous allez même pouvoir obtenir au sein de chaque palette des couleurs qui n'existent pas, comme l'orange, par exemple, obtenu en alternant les points de couleur rouge et jaune de la palette n° 0 ou le mauve, obtenu en alternant les points de couleur violet et blanc de la palette n° 1.

Pour cela, il vous sufit de dessiner deux traits l'un sur l'autre en alternant les points de couleur, le premier trait étant dessiné dans une couleur en débutant par un point éteint et le second dans une autre couleur en débutant par un point allumé. Bien entendu, il faut permuter les couleurs un trait sur deux, ce qui revient à utiliser à 4 reprises l'instruction **LINE**. Voici d'ailleurs à ce sujet un petit programme qui vous colore un rectangle en orange, quelle que soit d'ailleurs la couleur de fond : (noire dans le cas présent) :

Pour obtenir la couleur orange...

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 8,0
40 FOR Y=50 TO 150 STEP 2
50 LINE(80,Y)-(240,Y),2,,&H5555
60 LINE(80,Y)-(240,Y),3,,&HAAAA
70 LINE(80,Y+1)-(240,Y+1),2,,&HAAAA
80 LINE(80,Y+1)-(240,Y+1),3,,&H5555
90 NEXT
```

... il suffit de mélanger du rouge et du jaune !

Pour clore cette partie sur l'obtention d'un plus grand nombre de couleurs avec une même palette (la n° 0 en l'occurence...), voici un petit programme qui va se charger de dessiner, à la manière d'une mire, les 7 couleurs de l'arc-en-ciel (vous noterez l'encadrement en blanc, ce qui représente en fait 8 couleurs différentes) :

Cadre blanc de la mire...

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 9,0
40 FOR I=0 TO 5
50 LINE(I,I)-(319-I,199-I),3,B,&H5555
60 NEXT
70 X=6:COULEUR1=2:COULEUR2=2:GOSUB 140
80 X=50:COULEUR1=2:COULEUR2=3:GOSUB 140
90 X=94:COULEUR1=3:COULEUR2=3:GOSUB 140
100 X=138:COULEUR1=1:COULEUR2=1:GOSUB 140
110 X=182:COULEUR1=0:COULEUR2=1:GOSUB 140
120 X=270:COULEUR1=0:COULEUR2=2:GOSUB 140
130 END
140 FOR Y=6 TO 193 STEP 2
150 LINE(X,Y)-(X+43,Y),COULEUR1,,&H5555
160 LINE(X,Y)-(X+43,Y),COULEUR2,,&HAAAA
170 LINE(X,Y+1)-(X+43,Y+1),COULEUR1,,&HAAAA
180 LINE(X,Y+1)-(X+43,Y+1),COULEUR2,,&H5555
190 NEXT
200 RETURN
```

... suivi des 7 couleurs de l'arc-en-ciel

Sous-programme de dessin des bandes colorées

Il est possible d'obtenir sur une même image les couleurs des deux palettes en procédant par mélange.

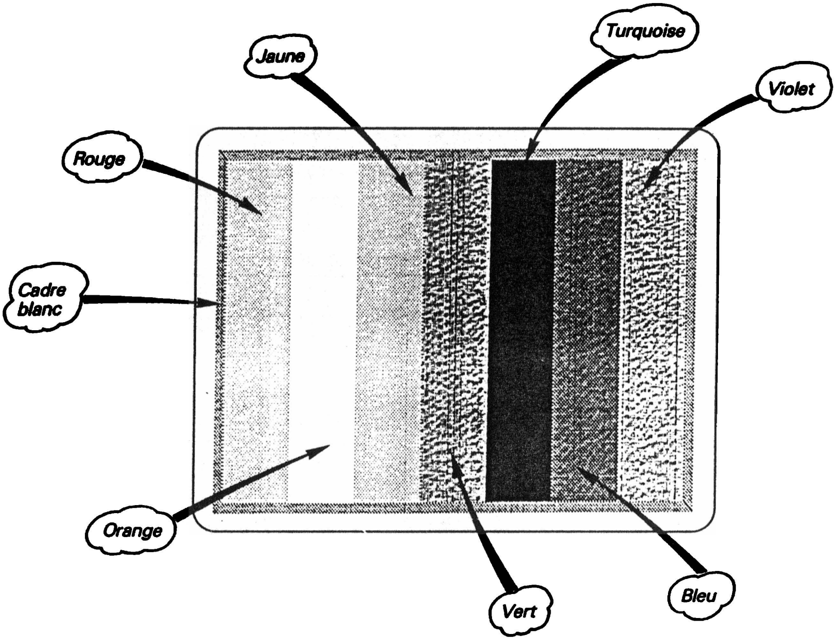


Fig. 3.7. : Mire de couleurs

DES COULEURS EN SUPPLÉMENT AVEC PAINT

PAINT
CHAINE DE CHARACTÈRES

Désormais, vous voilà en mesure de dessiner des traits dans au moins 8 couleurs différentes et ce, bien entendu,sur la même image. Peut-il en être de même avec l'instruction **PAINT** (*), à savoir, peut-on employer **PAINT** pour colorier des figures dans des couleurs autres que celle de la palette sélectionnée par **COLOR** ?...

La réponse à cette question est oui !... En effet, l'instruction **PAINT** présente une particularité intéressante (que nous allons exploiter à fond...) nous permettant de déterminer à notre guise la couleur, ainsi que l'allumage ou l'extinction de plusieurs points lumineux consécutifs et ce, grâce à une *chaîne de caractères* tenant guise de code de couleur. D'ailleurs, plutôt qu'un long discours, essayez donc ceci :

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 8,0
40 LINE(80,50)-(240,150),2,B
50 PAINT(160,100),CHR$(187)+CHR$(238),2
```

Chaîne de caractères remplaçant le code de la couleur de remplissage

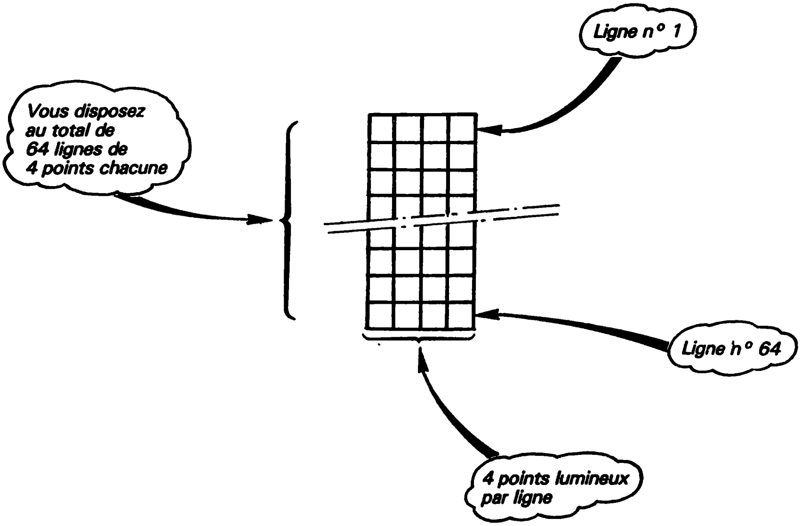
(*) La fiche détaillée de l'instruction **PAINT** figure à la page 323 du volume 2 de Pratique des IBM-PC et compatibles.

Que constatez-vous à l'exécution de ce programme ?... Qu'après avoir dessiné un rectangle rouge, celui-ci est ensuite colorié en *orange* !... En orange, c'est-à-dire dans une couleur qui n'existe pas !... Comment cela est-il possible ?

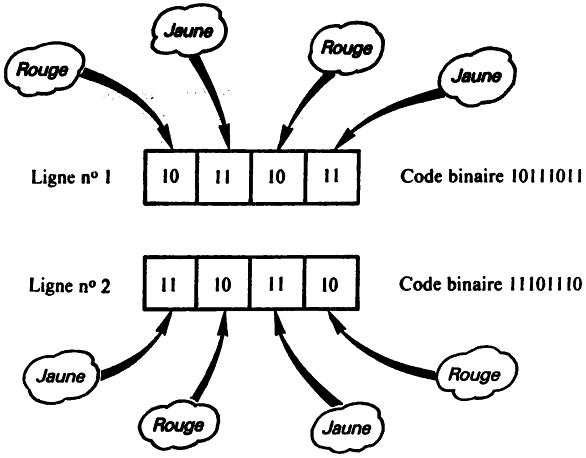
Pour exploiter ce petit « mystère », il faut que vous sachiez que la couleur de chaque point lumineux de l'écran graphique à moyenne résolution dépend de la valeur d'un code binaire à deux éléments binaires, code binaire que voici :

Code binaire	COULEUR	
	Palette n° 0	Palette n° 1
00	Couleur de fond	Couleur de fond
01	Vert	Turquoise
10	Rouge	Violet
11	Jaune	Blanc

Ceci étant, il faut que vous sachiez également que les points lumineux sur lesquels vous possédez tout pouvoir sont regroupés par lignes de 4 (par lignes de 8 en haute résolution) et ce, jusqu'à concurrence de 64 lignes.

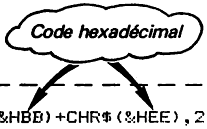


Dans ces conditions, il vous suffit de préciser pour chaque ligne utilisée la valeur de la combinaison binaire (exprimée en décimal ou en hexadécimal) qui résulte du choix des couleurs des 4 points de la ligne. Ainsi, si vous optez pour une alternance de points rouges et jaunes propres à produire de l'orange, et si vous décidez d'inverser une ligne sur deux la position des points rouges et jaunes, vous allez aboutir aux deux codes à 8 éléments binaires suivants :



Il vous suffit alors de convertir ces codes à 8 éléments binaires en décimal, ce qui donne 187 pour le premier et 238 pour le second, et à les introduire avec un **CHR$** en lieu et place du code de couleur de l'instruction **PAINT**.

Remarque : *Il vous est également possible de préciser les codes sous formes hexadécimale en rajoutant devant chacun d'eux les caractères &H. Ainsi, si l'on reprend l'exemple précédent, cela donne :*




```
50 PAINT(160,100),CHR$(&HBB)+CHR$(&HEE),2
```

Ainsi, grâce à cette particularité intéressante de l'instruction **PAINT**, vous voilà désormais à même de pouvoir colorier toute surface de votre choix avec plus de couleurs que prévu. D'ailleurs, pour vous y aider, voici la liste des codes à employer pour obtenir ces couleurs supplémentaires :

Couleurs supplémentaires	Chaîne de caractères	
	Forme décimale	Forme hexadécimale
Orange	CHR$ (187) + CHR$ (238)	CHR$ (&HBB) + CHR$ (&HEE)
Turquoise	CHR$ (17) + CHR$ (68)	CHR$ (&H11) + CHR$ (&H44)
Violet	CHR$ (34) + CHR$ (136)	CHR$ (&H22) + CHR$ (&H88)
Blanc	CHR$ (51) + CHR$ (204)	CHR$ (&H33) + CHR$ (&HCC)
Mauve (1)	CHR$ (187) + CHR$ (238)	CHR$ (&HBB) + CHR$ (&HEE)

(1) Cette couleur n'est accessible qu'avec la palette n° 1.

Il est possible de doubler le nombre de couleurs de coloriage de l'instruction PAINT en remplaçant le code de couleur par une chaîne de caractères.

LES COULEURS EN NOIR ET BLANC

Dessiner avec une palette de plus de 8 couleurs, c'est bien, mais en fait cela ne sert pas à grand chose !... En efet, plus de la moitié des utilisateurs d'IBM-PC et compatibles sont équipés de moniteurs vidéo monochrome pour qui un point sur l'écran, qu'il soit rouge, vert ou bleu, demeure un point. Il s'en suit que les différentes zones colorées se ressemblent toutes et que le magnifique diagramme en forme de camembert comportant 8 secteurs tous coloriés différemment que vous venez de terminer apparaîtra en vert (si le moniteur monochrome est vert, bien entendu...) sur fond vert. Autant dire qu'il sera totalement inexploitable sur tout autre moniteur qu'un moniteur couleurs.

Or, un bon programmeur se doit de réaliser un produit totalement compatible avec les différentes versions de moniteurs susceptibles d'être raccordés à un IBM-PC. Sans cela, son produit est condamné à une diffusion plus que restreinte !

Mais ce n'est pas tout !... A quoi cela sert-il d'afficher un graphique à l'écran, si ce n'est pour en faire une copie sur papier ?... Or, en matière d'imprimante, plus de 95 % des utilisateurs ne disposent que d'un modèle noir et blanc classique. Dans ces conditions, chaque point de couleur est reproduit par un point noir et le magnifique graphique en couleurs que vous cherchez à imprimer ne donne lieu qu'à une immense tache sombre.

C'est donc pour vous éviter ces quelques « désagréments » de nature à décourager les bonnes volontés, qu'il convient d'aborder le coloriage sous un autre aspect : celui de la densité de points de couleurs. Toutefois, avant d'aller plus loin dans cette voie, essayez donc le petit programme que voici :

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 8,0
40 LINE (0,66)-(319,133),2,B
50 FOR X=40 TO 280 STEP 40
60 LINE(X,66)-(X,133),2
70 NEXT
80 FOR I=1 TO 8
90 READ A,B,C,D
100 DENSITE$=CHR$(A)+CHR$(B)+CHR$(C)+CHR$(D)
110 PAINT (I*40-20,100),DENSITE$,2
120 NEXT
130 DATA 64,0,4,0
140 DATA 64,4,64,4
150 DATA 64,17,4,17
160 DATA 68,17,68,17
170 DATA 21,68,81,68
180 DATA 21,81,21,81
190 DATA 21,85,81,85
200 DATA 85,85,85,85
```

Dessin de 8 boîtes... (lignes 40 à 70)

... que l'on remplit de points de couleurs (ligne 110)

Chaîne de caractères qui détermine la densité des points

... qui, une fois exécuté, produit l'affichage que voici :

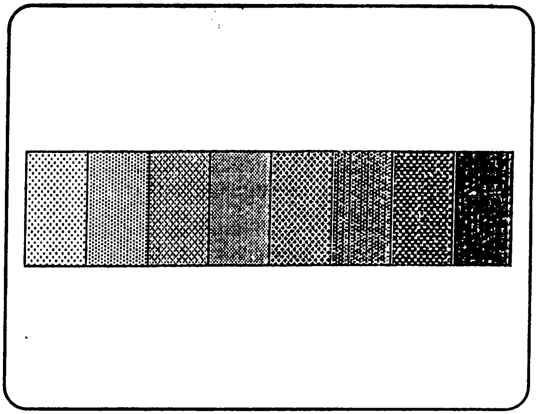


Fig. 3.8. : Une même couleur reproduite avec huit densités de points différentes.

Que constatez-vous ?... Que cette fois-ci la même couleur, le vert en l'occurence, est reproduit avec huit densités de points différentes, allant de 1/8)1 point allumé sur 8) jusqu'à 8/8 (tous les points allumés) ce qui, pour ceux qui sont équipés de moniteurs monochrome, donne lieu à une belle gamme de dégradés de gris. Génial, non ?

Et ne croyez pas que pour autant cela limite l'emploi des couleurs à une seule !... Vous pouvez très bien combiner l'alternance des couleurs associées à des densités de points différentes pour obtenir à la fois des images colorées qui aient de l'allure tant sur l'écran d'un moniteur que reproduites en noir et blanc sur papier. Pour cela, il vous suffit de consulter la table ci-dessous qui vous précise les codes à employer pour obtenir une densité de points donnée dans une couleur donnée :

Couleur	Densité de points						
	1/8	1/4(*)	3/8	1/2(*)	5/8	3/4(*)	7/8
Vert ou Turquoise	64 0 4 0	64 4	64 17 4 17	68 17	21 68 81 68	21 81	21 85 21 85
Rouge ou Violet	128 0 8 0	128 8	128 34 8 34	136 34	42 136 162 136	42 162	42 170 42 170
Jaune ou Blanc	192 0 12 0	192 12	192 51 12 51	204 51	63 204 243 204	63 243	63 255 63 255

(*) Ne nécessitent que 2 codes au lieu de 4.

Remarque : *Vous noterez que les codes des densités des points des couleurs rouge, violet, jaune et vert s'obtiennent en multipliant par 2 ou par 3 ceux des couleurs vert et turquoise. Cela peut toujours servir !*

Vous voulez un exemple concret de graphique réalisé dans cette optique ?... Qu'à cela ne tienne, en voici un qui va vous dessiner votre camembert préféré en 3 couleurs et 6 densités différentes :

```
10 KEY OFF
20 SCREEN 1:CLS
30 COLOR 8,0
40 PI=3.141593
50 XO=160:YO=100:RAYON=100
60 FOR I=1 TO 6
70 READ DEGRES,DECALAGE,CODE1,CODE2,CODE3,CODE4
80 RADIANS=DEGRES*PI/180
90 FIN=DEBUT+RADIANS
100 BISSECTRICE=(FIN+DEBUT)/2
110 DX=DECALAGE*COS(BISSECTRICE)
120 DY=DECALAGE*SIN(BISSECTRICE)*5/6
130 CIRCLE (XO+DX,YO-DY),RAYON,3,-DEBUT,-FIN
140 DX=RAYON/2*COS(BISSECTRICE)
150 DY=RAYON/2*SIN(BISSECTRICE)*5/6
160 DENSITE$=CHR$(CODE1)+CHR$(CODE2)+CHR$(CODE3)+CHR$(CODE4)
170 PAINT(XO+DX,YO-DY),DENSITE$,3
180 DEBUT=FIN
190 NEXT
200 DATA 45,25,64,4,64,4
210 DATA 75,0,42,136,162,136
220 DATA 30,0,192,0,12,0
230 DATA 90,0,21,81,21,81
240 DATA 45,0,128,34,8,34
250 DATA 75,0,204,51,204,51
```

Lecture des données de chaque «portion» du camembert (ligne 70)

Dessin des portions.... (ligne 130)

... et leur remplissage de points colorés ! (ligne 170)

Pas mal, n'est-ce pas ?... Comme vous pouvez le constater de visu, vous obtenez un beau diagramme circulaire composé de 6 secteurs alternativement dessinés dans les 3 couleurs vert, rouge et jaune avec pour densités de points, en commençant par celui qui est décalé : 1/4, 5/8, 1/8, 3/4, 3/8 et 1/2.

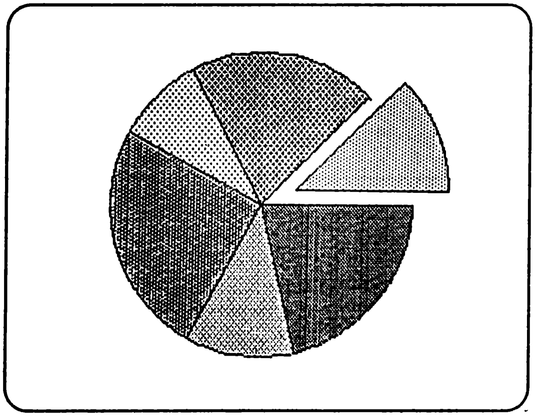


Fig. 3.9. : Camembert dont les « portions » sont à la fois de couleurs et de densités de points différentes.

Pour être compatibles en couleurs comme en noir et blanc, les figures doivent être coloriées en dosant la densité des points de couleurs.

COLORIAGE EN HAUTE RÉSOLUTION

Pour aller encore un peu plus loin dans la voie du coloriage, il nous faut maintenant aborder celui-ci en mode graphique à haute résolution. Mais d'abord, pourquoi vouloir se servir du mode graphique à haute résolution ?... Présente-t-il un intérêt quelconque ?
conque ?

La réponse à de telles questions passe par l'analyse du nombre maximal de caractères par ligne que nécessitent vos graphiques. Si ces derniers ne nécessitent aucun commentaire ou si ceux-ci peuvent tenir à raison de 40 caractères par ligne, alors il n'y a pas lieu de vouloir quitter le mode graphique à moyenne résolution. A moins, bien sûr, que ce soient les graphiques eux-mêmes qui réclament une résolution plus fine !

Par contre, si les commentaires que vous souhaitez apporter à vos graphiques ne peuvent se satisfaire des « grosses » lettres du mode graphiques à moyenne résolution (c'est le cas de la quasi-totalité des graphiques de gestion...), alors il vous faut aborder le mode graphique à haute résolution avec ce qu'il comporte d'avantages (80 caractères/lignes, résolution horizontale double) et d'inconvénients (perte de la couleur, coloriage uniforme).

Dans ces conditions, la seule possibilité qui s'offre à vous pour distinguer les zones les unes des autres consiste à les colorier avec des densités de points différentes. Toutefois, pour satisfaisante qu'elle soit, cette solution peut encore être améliorée par la création de motifs géométriques répétitifs tels que hachures, damiers, trames, etc, et cela, d'autant plus facilement que le contrôle de l'allumage ou de l'extinction des points s'exerce désormais sur 8 points au lieu de 4.

A titre d'encouragement à poursuivre dans cette voie, essayez donc les quelques lignes que voici :

```
10 KEY OFF
20 SCREEN 2:CLS
30 FOR Y=0 TO 134 STEP 67
40 FOR X=0 TO 483 STEP 161
50 READ CODE1,CODE2,CODE3,CODE4,CODE5,CODE6,CODE7,CODE8
60 LINE(X,Y)-(X+156,Y+65),1,B
70 MOTIF$=CHR$(CODE1)+CHR$(CODE2)+CHR$(CODE3)+CHR$(CODE4)+CHR$(CODE5)+CHR$(CODE6
)+CHR$(CODE7)+CHR$(CODE8)
80 PAINT (X+1,Y+1),MOTIF$
90 NEXT X,Y
100 DATA 255,255,255,255,255,255,255,255
110 DATA 192,48,12,3,192,48,12,3
120 DATA 192,0,12,0,192,0,12,0
130 DATA 240,240,15,15,240,240,15,15
140 DATA 136,128,156,128,136,8,201,8
150 DATA 12,51,192,192,192,51,12,12
160 DATA 51,51,51,51,51,51,51,51
170 DATA 3,12,48,192,3,12,48,192
180 DATA 1,199,1,0,16,124,16,0
190 DATA 3,15,63,0,3,15,63,0
200 DATA 207,48,252,3,207,48,252,3
210 DATA 3,3,3,252,3,3,3,252
```

Codes des 12 motifs géométriques

Vous devriez obtenir (mais sait-on jamais...) les 12 motifs géométriques suivants :

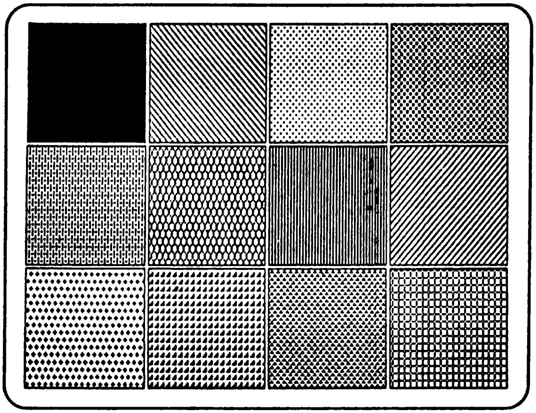
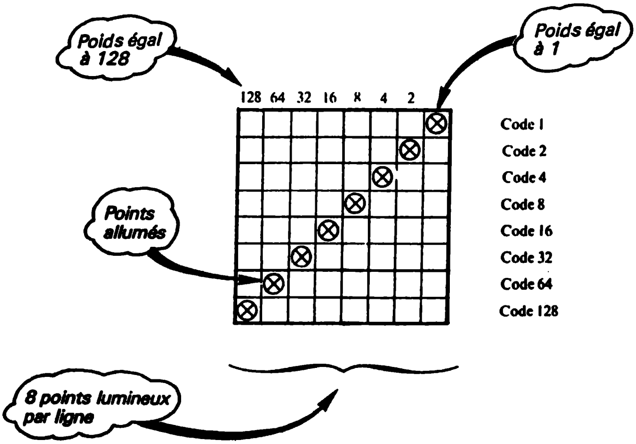


Fig. 3.10. : Quelques motifs géométriques parmi tant d'autres.

Comme vous pouvez le constater à la frappe de la liste des données, ce ne sont plus 4 mais 8 codes que les motifs géométriques nécessitent et ce, pour prendre en compte les 8 points lumineux par ligne du mode graphique à haute résolution. En effet, si l'on prend l'exemple suivant qui représente des hachures obliques et si, sachant que chaque point allumé correspond à un « 1 » (puisqu'il n'y a plus de couleur...) et chaque point éteint à un « 0 », on aboutit à la grille de codage que voici :



Remarque : *Selon le motif que vous cherchez à obtenir, il se peut que certains codes soient égaux à 0 par absence de points sur la ligne. Sachez dans ce cas que l'instruction PAINT n'accepte pas plus de deux zéros consécutifs. Au-delà, vous n'obtenez que le message d'erreur ILLEGAL FUNCTION CALL.*

Avant de quitter ce chapitre sur le coloriage, voici un petit essai sans prétention destiné à illustrer l'emploi des motifs géométriques dans les graphiques de gestion :

```
10 KEY OFF
20 SCREEN 2:CLS
30 FOR I=1 TO 4
40 READ CODE1,CODE2,CODE3,CODE4
50 MOTIF$(I)=CHR$(CODE1)+CHR$(CODE2)+CHR$(CODE3)+CHR$(CODE4)
60 NEXT
70 LINE(0,180)-(639,180),1
80 FOR J=1 TO 6
90 X=J*100-60:Y=180
100 FOR I=1 TO 4
110 READ HAUTEUR
120 LINE(X,Y)-(X+60,Y-HAUTEUR),1,B
130 IF HAUTEUR>0 THEN PAINT(X+1,Y-1),MOTIF$(I)
140 Y=Y-HAUTEUR
150 NEXT I,J
160 DATA 255,255,255,255
170 DATA 192,48,12,3
180 DATA 192,0,12,0
190 DATA 240,240,15,15
200 DATA 10,10,5,0
210 DATA 15,15,10,5
220 DATA 20,20,15,10
230 DATA 25,25,20,20
240 DATA 30,30,25,30
250 DATA 35,35,30,40
```

Chargement et mise en tableau des motifs (lignes 10 à 60)

Dessin et coloriage du graphique en barres superposées (lignes 70 à 150)

Codes des 4 motifs (lignes 160 à 190)

Valeurs numériques à dessiner sous forme de barres superposées (lignes 200 à 250)

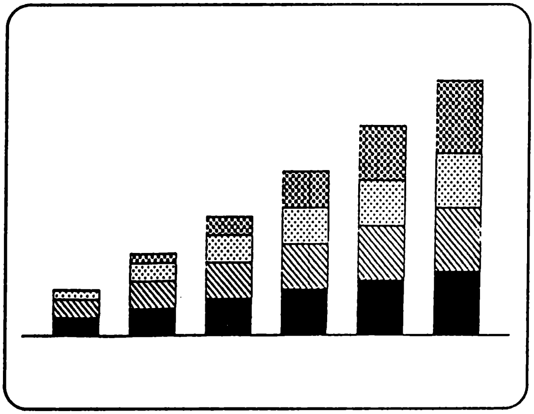


Fig. 3.11. : Essai de coloriage en noir et blanc.

Intéressant, n'est-ce pas !... Toutefois, il vous faudra attendre le chapitre 5 pour en savoir plus sur l'art et la manière de dessiner ce type de graphiques !

> **Le coloriage en haute résolution s'opère par l'emploi de motifs géométriques répétitifs.**

Exercice sur le chapitre 3

1. Peut-on colorier l'intérieur d'un rectangle avec l'instruction LINE ?

   □ Oui, à condition d'employer l'option F.
   □ Non, il faut pour cela utiliser l'instruction FILL.

2. Dans quel cas l'instruction PAINT ne sait-elle pas reconnaître les limites de la figure à colorier ?

   □ Lorsque la couleur intérieure est différente de celle du pourtour.
   □ Lorsque la couleur du pourtour n'est pas indiquée.

3. Est-il possible de dessiner un trait en orange ?

   □ Non, car cette couleur n'existe pas.
   □ Oui, à condition d'alterner les points rouges et jaunes.

4. Peut-on colorier en violet une figure dont le contour est vert ?

   □ Cela ne se peut pas car le violet et le vert font partie de deux palettes différentes.
   □ Oui, à condition d'employer un fond bleu et de n'allumer en rouge qu'un point sur deux.

5. Que permet d'obtenir la chaîne de caractères CHR$ (42) + CHR$ (162) placée à la suite de l'instruction PAINT ?

   □ La couleur rouge avec une densité de points de 3/4.
   □ La couleur jaune avec une densité de points de 1/4.

6. Pourquoi est-il plus facile d'obtenir des motifs géométriques de coloriage en haute résolution qu'en moyenne ?

   □ Parce que ce mode graphique est en noir et blanc.
   □ Parce que vous disposez de deux fois plus de points pour les coder.

(Réponses page 78)

Solutions des exercices sur le chapitre 1

1. 640 × 200
2. Oui, pour pallier au cas où l'instruction SCREEN « oublie » d'effacer l'écran.
3. Le rouge saturé à 100 %
4. 4
5. PSET (bien que cela soit également possible avec PRESET)
6. Cela dépend de la position du dernier point tracé
7. A indiquer au basic que les coordonnées graphiques sont exprimées en valeurs relatives, ainsi qu'à déterminer le pas d'incrémentation d'une boucle.

CHAPITRE IV

REPRÉSENTATION GRAPHIQUE DES DONNÉES

*La représentation graphique des données est l'un des thèmes majeurs de l'art graphique. Et ne croyez pas pour autant que cela s'accompagne de difficultés insurmontables !... En fait, les connaissances acquises au cours des trois précédents chapitres suffisent largement pour aborder et maîtriser un tel sujet avec facilité, à tel point qu'après la lecture des deux chapitres à venir, vous serez à même de personnaliser le logiciel **DATAGRAF** de représentation graphique des données qui y est expliqué par le détail.*

Menu du chapitre 4

Analyse du problème posé par la représentation graphique des données.
Elaboration du logiciel DATAGRAF.
Ecran du menu principal.
Ecran d'entrée des données.
Sous-programme d'INPUT contrôlé.
Ecran de sauvegarde et de chargement des données.
Ecran de sélection d'un diagramme.

Quel est votre choix : ■

ANALYSE DU PROBLÈME POSÉ PAR LA REPRÉSENTTION GRAPHIQUE

Il ne faudrait pas croire que la simple possession de données numériques suffise à produire un diagramme représentatif de celles-ci tel que celui figurant ci-après (qui n'est rien d'autre qu'un diagramme à colonnes empilées) :

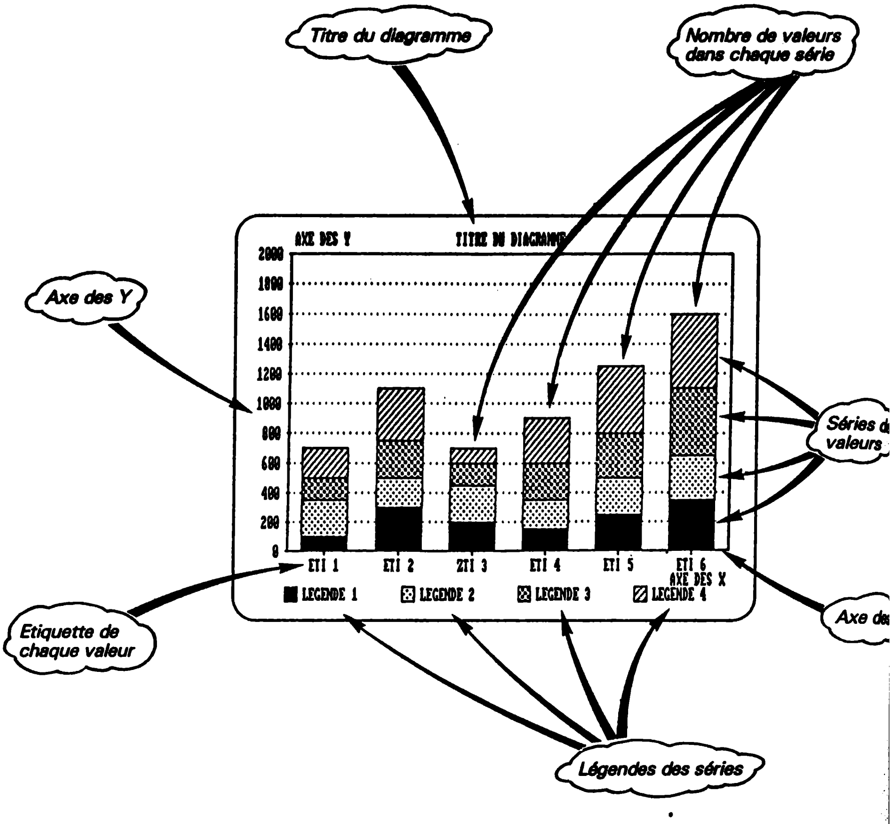


Fig. 4.1. : Diagramme à colonnes empilées.

Comme vous pouvez le constater, la connaissance des valeurs des données s'accompagne d'un grand nombre de paramètres propres à permettre, précisément, l'interprétation de celles-ci. Ce sont notamment :

* Le *titre du diagramme* sans lequel toute analyse des données s'avère problématique dans la mesure où, dès le départ, l'on ne sait à quoi elles se rapportent.

* Les *axes des X et des Y* dont la désignation et les échelles graduées permettent d'apprécier la valeur absolue des données selon deux critères distincts (généralement l'amplitude et le temps).

* Le nombre de *séries de valeurs* ainsi que le nombre de *valeurs dans chaque série* de façon à ce que l'ordinateur sache avec précision où placer chacune des données que vous lui demandez d'afficher.

* Les *légendes* des diverses séries de valeurs dont l'indication permet d'effectuer des comparaisons relatives entre séries en toute connaissance de cause.

* Les *étiquettes* rattachées à chacune des valeurs sans lesquelles le diagramme demeure sans signification.

* La *nature du diagramme* que l'on souhaite obtenir. Ce peut être un diagramme à traits continus, à aires superposées, à colonnes (empilées ou non) et à secteurs circulaires (plus communément appelés « camembert »).

Ceci étant, il faut également pouvoir mémoriser le fruit de votre travail (taper une suite de valeurs numériques, c'est long...), non sous la forme de graphiques, mais bien plutôt sous la forme de fichier de données. Cela prend moins de place en mémoire qu'une image et il suffit d'une commande (enfin presque...) pour que l'ordinateur vous redessine quasi-instantanément le diagramme sous vos yeux.

Enfin, il faut prévoir l'impression sur papier du diagramme reproduit par l'écran, non pas tant en ce qui concerne le programme de recopie d'écran (les touches [Ctrl]—[PrtSc] y pourvoient aisément, mais bien plutôt sous l'aspect où l'impression se faisant en noir et blanc, l'image élaborée par le programme se doit d'être conforme à ce critère essentiel (hachures et motifs géométriques à la place des couleurs, caractères au format 80 colonnes, etc, etc...).

Un diagramme de représentation graphique des données nécessite que soient pris en compte un certain nombre de paramètres et de critères essentiels.

ÉLABORATION DU LOGICIEL GRAPHIQUE DATAGRAF

Fort de l'analyse précédente, nous allons élaborer ensemble un logiciel « minimum » de représentation graphique des données baptisé DATAGRAF pour la circonstance. Pourquoi « minimum » ?... D'abord, pour qu'il ne soit pas trop long à taper et qu'ensuite, il soit suffisamment « ouvert » pour que vous puissiez y incorporer vos innovations personnelles !

Tout d'abord, nous allons limiter à 4 le nombre d'écrans de menu. Ce seront pour la circonstance :

* L'écran du menu principal.
* L'écran d'entrée des données.
* L'écran de sauvegarde et de chargement des données.
* L'écran de sélection d'un diagramme.

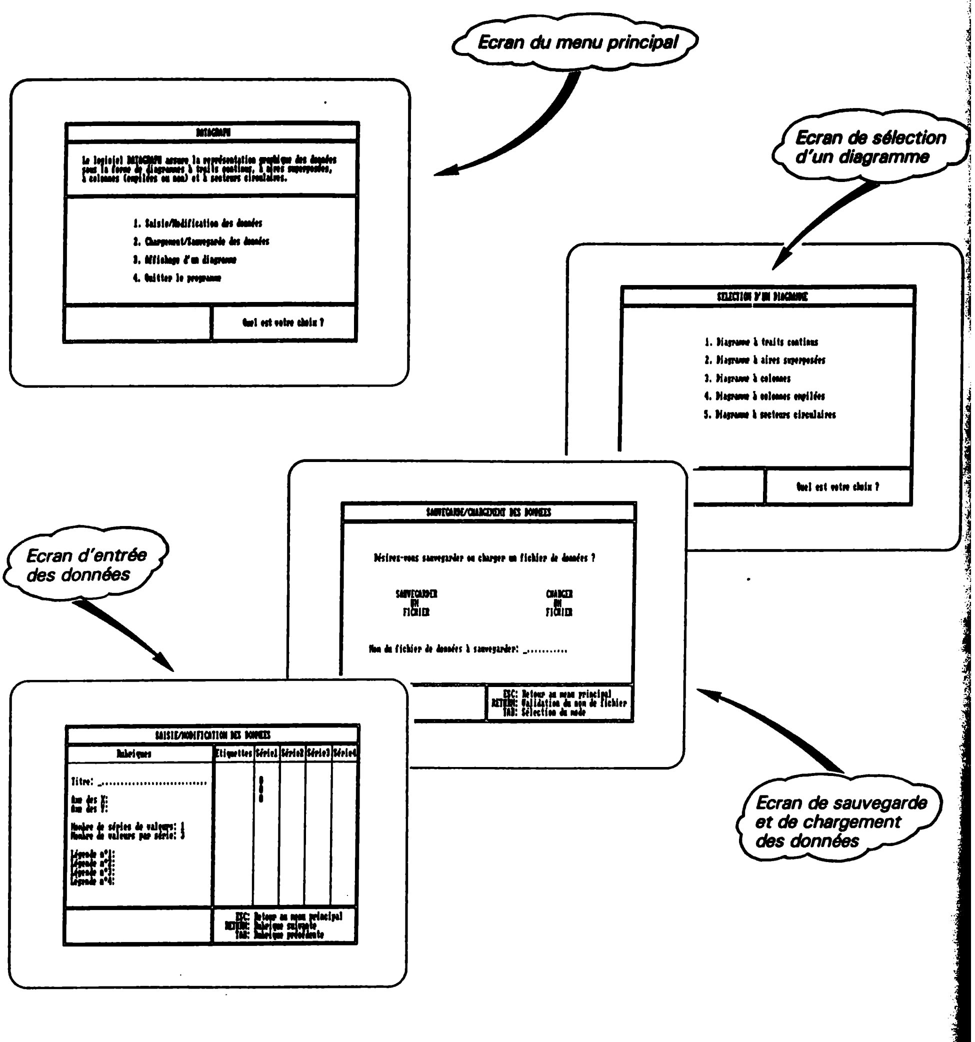


Fig. 4.2. : Différents écrans du logiciels DATAGRAF

Ensuite, c'est le nombre de séries et de valeurs dans chaque série que nous allons limiter. En l'occurence, nous ne pourrons pas dépasser 4 séries de valeurs, chaque série pouvant comporter de 3 (minimum) à 12 (maximum) valeurs distinctes. Bien sûr, rien ne vous oblige à en rester là si vos besoins spécifiques nécessitent plus de séries ou plus de valeurs dans chacune des séries.

Quatre séries de douze valeurs vous semblent peu ?... Alors, jetez donc un coup d'œil au diagramme ci-dessous qui est un diagramme à colonnes comportant précisément 4 séries de 12 valeurs :

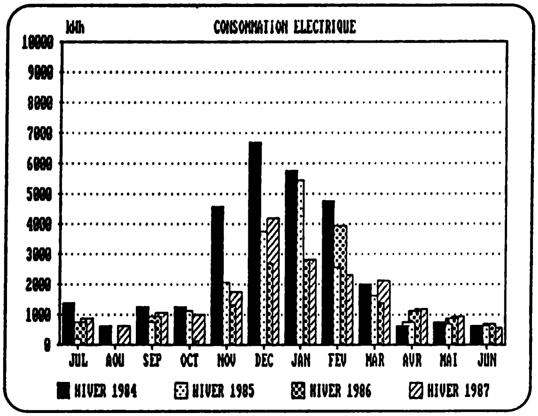


Fig. 4.3. : Diagramme à colonnes comportant 4 séries de 12 valeurs.

Il paraît difficile d'en loger plus dans un si petit espace, vous ne trouvez pas ?... Toutefois, tout est toujours possible et rien ne s'oppose à une concentration plus élevée si tel est votre bon plaisir.

Enfin, c'est une sélection des 5 diagrammes les plus couramment utilisés qui vous est proposée, à savoir :

* Les diagrammes à traits continus

* Les diagrammes à aires superposées

* Les diagrammes à colonnes

* Les diagrammes à colonnes empilées

* Les diagrammes à secteurs circulaires

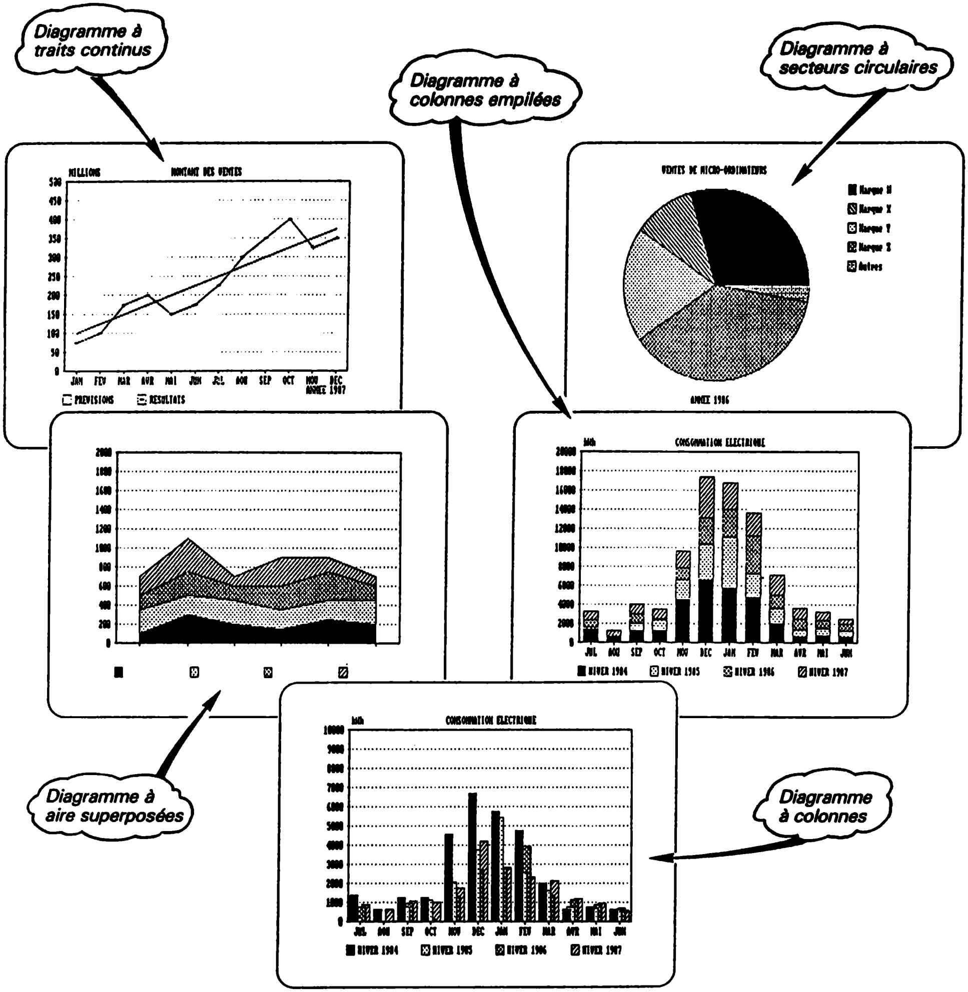


Fig. 4.4. : Différents diagrammes dessinés par le logiciel DATAGRAF.

Or, c'est précisément au niveau des diagrammes que vous risquez de ne pas trouver chaussure à votre pied et de devoir modifier, voire créer celui qui vous convient très exactement. En cela, vous serez aidé par les descriptions détaillées qui en sont données au niveau du chapitre suivant. Il y en a forcément un qui s'apparente à celui que vous cherchez à obtenir et, dans ce cas, il vous suffira de vous en inspirer fortement (ils sont là pour cela...).

Le logiciel DATAGRAF qui va être décrit permet de représenter sous forme de 5 diagrammes différents un maximum de 4 séries de 12 valeurs numériques.

ÉCRAN DU MENU PRINCIPAL

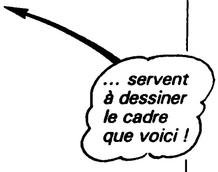
Vous voici arrivé au pied du mur !... Si donc vous vous sentez d'attaque pour cette aventure, respirez un grand coup, faites jouer vos articulations des doigts et en avant !

Pour débuter, voici quelques chaînes destinées à dessiner des traits sur l'écran texte, assorties d'une ligne de dimensionnement des tableaux et des variables indicées qu'emploie le programme, ainsi que d'une ligne d'affectation de valeurs par défaut aux variables essentielles au bon déroulement du programme. En l'occurence, il s'agit des variables SERIES (nombre de séries de valeurs), VALEURS (nombre de valeurs dans chaque série) et DIAGRAMME (numéro du diagramme à afficher) auxquelles on attribue respectivement les valeurs 1, 3 et 1 :

```
10 KEY OFF
20 DIM AXE$(2),ETIQUETTE$(12),DONNEE(12,4),CARACTERE$(80)
30 TRAIT1$=CHR$(201)+STRING$(77,205)+CHR$(187)
40 TRAIT2$=CHR$(186)+STRING$(77,32)+CHR$(186)
50 TRAIT3$=CHR$(204)+STRING$(77,205)+CHR$(185)
60 TRAIT4$=CHR$(204)+STRING$(38,205)+CHR$(203)+STRING$(38,205)+CHR$(185)
70 TRAIT5$=CHR$(186)+STRING$(38,32)+CHR$(186)+STRING$(38,32)+CHR$(186)
80 TRAIT6$=CHR$(200)+STRING$(38,205)+CHR$(202)+STRING$(38,205)+CHR$(188)
90 TRAIT7$=CHR$(204)+STRING$(38,205)+CHR$(203)+STRING$(10,205)+
   CHR$(209)+STRING$(6,205)+CHR$(209)+STRING$(6,205)+CHR$(209)+
   STRING$(6,205)+CHR$(209)+STRING$(6,205)+CHR$(185)
100 TRAIT8$=CHR$(186)+STRING$(38,32)+CHR$(186)+STRING$(10,32)+
    CHR$(179)+STRING$(6,32)+CHR$(179)+STRING$(6,32)+CHR$(179)+
    STRING$(6,32)+CHR$(179)+STRING$(6,32)+CHR$(186)
110 TRAIT9$=CHR$(199)+STRING$(38,196)+CHR$(215)+STRING$(10,196)+
    CHR$(197)+STRING$(6,196)+CHR$(197)+STRING$(6,196)+CHR$(197)+
    STRING$(6,196)+CHR$(197)+STRING$(6,196)+CHR$(182)
120 TRAIT10$=CHR$(204)+STRING$(38,205)+CHR$(206)+STRING$(10,205)+
    CHR$(207)+STRING$(6,205)+CHR$(207)+STRING$(6,205)+CHR$(207)+
    STRING$(6,205)+CHR$(207)+STRING$(6,205)+CHR$(185)
130 SERIES=1:VALEURS=3:DIAGRAMME=1
```

Comme tout ceci ne donne aucun résultat visible à l'écran, (tout du moins pas pour le déroulement...), il vous faut taper la suite du programme du menu principal pour qu'apparaîssent vos (éventuelles...) fautes de frappes. Voici donc cette suite :

```
140 'Menu principal-----------------------------------------------------------
150 COLOR 15,1:CLS:PRINT TRAIT1$:PRINT TRAIT2$:PRINT TRAIT3$
160 FOR I=1 TO 5:PRINT TRAIT2$:NEXT:PRINT TRAIT3$
170 FOR I=1  TO 11:PRINT TRAIT2$:NEXT:PRINT TRAIT4$
180 FOR I=1 TO 3:PRINT TRAIT5$;:NEXT:LOCATE 25,1:PRINT TRAIT6$;
190 COLOR 14:LOCATE 2,36:PRINT"DATAGRAPH"
200 COLOR 13:LOCATE 5,6:PRINT"Le logiciel DATAGRAPH ";
    "assure la représentation graphique des données"
210 LOCATE 6,6:PRINT"sous la forme de diagrammes ";
    "à traits continus, à aires superposées,"
220 LOCATE 7,6:PRINT"à colonnes (empilées ";
   "ou non) et à secteurs circulaires."
230 COLOR 11:LOCATE 12,20:PRINT"1. Saisie/Modification des données"
240 LOCATE 14,20:PRINT"2. Chargement/Sauvegarde des données"
250 LOCATE 16,20:PRINT"3. Affichage d'un diagramme"
260 LOCATE 18,20:PRINT"4. Quitter le programme"
270 COLOR 10:LOCATE 23,49:PRINT"Quel est votre choix ?"
280 CHOIX$=INPUT$(1):CHOIX=VAL(CHOIX$)
290 COLOR 15:LOCATE 23,71:PRINT CHOIX
300 GOSUB 1300:IF ERREUR<>0 THEN 280
310 ON CHOIX GOTO 340,830,1150:COLOR 15,0:CLS:KEY ON:END
```



Ceci étant, vous devriez obtenir le beau menu principal que voici :

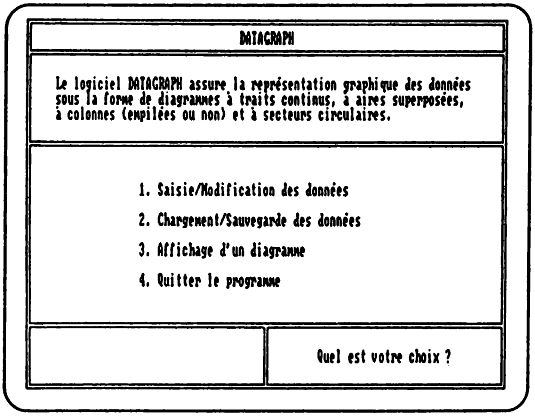


Fig. 4.5. : Ecran du menu principal.

Voyons maintenant un peu comment tout cela fonctionne !... Après avoir supprimé l'affichage des touches de fonction (ligne 10), le programme procède alors au dimensionnement (ligne 20) des variables **AXE$** (-) des axes X et Y, **ETIQUETTES** (-) de chacune des valeurs, **DONNEE** (-,-) des 4 séries de 12 valeurs et **CARACTERE$** (-) dont on verra le rôle dans très peu de temps.

Ceci fait, le programme se charge alors de déclarer les chaînes **TRAIT1$** à **TRAIT10$** destinées à simplifier la vie du programmeur qui aime placer des cadres autour de ses menus (la simplification opérant une fois les chaînes tapées, bien entendu...). Puis c'est au tour de la ligne 130 d'affecter leurs valeurs par défaut aux variables **SERIES**, **VALEURS** et **DIAGRAMME**.

Tout ceci n'étant qu'entrée en matière, passons au dessin du masque d'écran du menu principal. Tout d'abord, c'est un cadre à double traits blancs sur fond bleu que les lignes 150 à 180 dessinent autour de l'écran. Puis, c'est au tour des lignes 190 à 270 d'afficher le titre ainsi que les différentes rubriques de ce menu.

Variable	Désignation
AXE$ (-)	Désignation des axes X et Y
CARACTERE$ (-)	Voir sous-programme INPUT
CHOIX	Choix de l'option (de 1 à 4)
CHOIX$	Variable chaîne du choix
DIAGRAMME	Numéro du diagramme sélectionné (1 à 5)
DONNÉE (-,-)	Donnée numérique (4 séries de 12 valeurs)
ERREUR	Vaut 1 quand il y a une erreur
ETIQUETTE$ (-)	Etiquettes des valeurs (de 1 à 4)
SÉRIES	Séries de valeurs (de 1 à 4)
TITRE$	Titre du diagramme
TRAIT-$	Traits du cadre (il y en a 10 au total)
VALEURS	Nombre de valeurs par série (de 1 à 12)

Vous noterez à la ligne 270 l'emploi d'un PRINT au lieu d'un INPUT pour vous demander quel choix est le vôtre.

Pourquoi cela ?... Mais pour vous éviter d'avoir à appuyer sur la touche [RETURN] après avoir frappé un chiffre (moins on en fait...). C'est pourquoi vous allez trouver à la ligne 280 un INPUT$ (1) qui, comme sa syntaxe le laisse supposer, n'attend de vous qu'un seul chiffre... et surtout pas un caractère alphabétique !... De toute façon, la conversion de la chaîne CHOIX$ en la valeur numérique CHOIX ne laisse aucune chance de passer aux lettres et signes de ponctuation, l'ensemble donnant lieu à un zéro.

Maintenant, à quoi peut bien correspondre le sous-programme 1300 ?... Mais au sous-programme de traitement des messages d'erreur que voici :

C'est ici qu'on vérifie si vous avez tapé le bon numéro...

```
1290 'Traitement des messages d'erreurs-----------------------------------
1300 IF CHOIX>=1 AND CHOIX<=4 THEN 1540
1310 BEEP:COLOR 12
1320 LOCATE 22,10:PRINT"Seules les valeurs"
1330 LOCATE 23,10:PRINT"1 à 4 sont autorisées."
1340 LOCATE 24,10:PRINT"Veuillez recommencer !";
1350 ERREUR=1:COLOR 15:RETURN
1360 IF SERIES>=1 AND SERIES<=4 THEN 1540
1370 BEEP:COLOR 12
1380 LOCATE 22,10:PRINT"Seules les valeurs"
1390 LOCATE 23,10:PRINT"1 à 4 sont autorisées."
1400 LOCATE 24,10:PRINT"Veuillez recommencer !";
1410 ERREUR=1:COLOR 15:RETURN
1420 IF VALEURS=3 OR VALEURS=4 OR VALEURS=6 OR VALEURS=12 THEN 1540
1430 BEEP:COLOR 12
1440 LOCATE 22,10:PRINT"Seules les valeurs 3 4"
1450 LOCATE 23,10:PRINT"6 et 12 sont autorisées."
1460 LOCATE 24,10:PRINT"Veuillez recommencer !";
1470 ERREUR=1:COLOR 15:RETURN
1480 IF DIAGRAMME>=1 AND DIAGRAMME<=5 THEN 1540
1490 BEEP:COLOR 12
1500 LOCATE 22,10:PRINT"Seules les valeurs"
1510 LOCATE 23,10:PRINT"1 à 5 sont autorisées."
1520 LOCATE 24,10:PRINT"Veuillez recommencer !";
1530 ERREUR=1:COLOR 15:RETURN
1540 LOCATE 22,2:PRINT SPC(38):LOCATE 23,2:PRINT SPC(38)
1550 LOCATE 24,2:PRINT SPC(38):ERREUR=0:RETURN
```

... et c'est ici qu'on efface le message d'erreur que vous avez obtenu !

Comme vous pouvez le constater, il ne sert pas seulement au menu principal. Il y a également les 3 autres écrans qui ne se gènent pas pour l'utiliser !... Toutefois, en ce qui concerne le menu principal, seules les lignes 1300 à 1350 et 1540 à 1550 sont à prendre en considération. En effet, la ligne 1300 commence par vérifier si le chiffre que vous avez tapé est compris entre 1 et 4. Si c'est le cas, elle vous expédie vers les lignes 1540 et 1550 qui assurent l'effacement de la fenêtre des messages d'erreur (en bas à gauche du menu). Si par contre ce n'est pas le cas, vous obtenez alors un bip sonore (en général, c'est toujours mauvais signe...) suivi du message d'erreur approprié (lignes1310 à 1340). Notez qu'avant de vous réexpédier d'où vous venez, la ligne 1350 met à 1 la variable **ERREUR**, ce que la ligne 300 du programme du menu principal met aussitôt à profit pour réitérer la question du choix. Ce n'est que lorsque ce dernier est enfin correctement accompli que la variable **ERREUR** vaut 0, ce qui vous autorise enfin à accéder à la ligne 310 d'aiguillage vers les autres écrans. Quant à la fin de cette ligne, c'est elle qui vous permet de sortir du programme sans passer par le [CTRL] – [BREAK] habituel.

Pour contrôler le choix d'une option, le plus simple consiste à transférer dans une chaîne le contenu de INPUT$ (1) et à convertir ensuite cette chaîne en un nombre avec VAL.

ÉCRAN D'ENTRÉE DES DONNÉES

Au tour maintenant de l'écran d'entrée des données. Comme vous pouvez vous en douter, ce n'est pas le plus simple à réaliser et ce, d'autant plus qu'une indication en bas de cet écran laisse entendre qu'il est possible de progresser d'avant en arrière à travers les rubriques grâce aux touches RETURN et TAB.

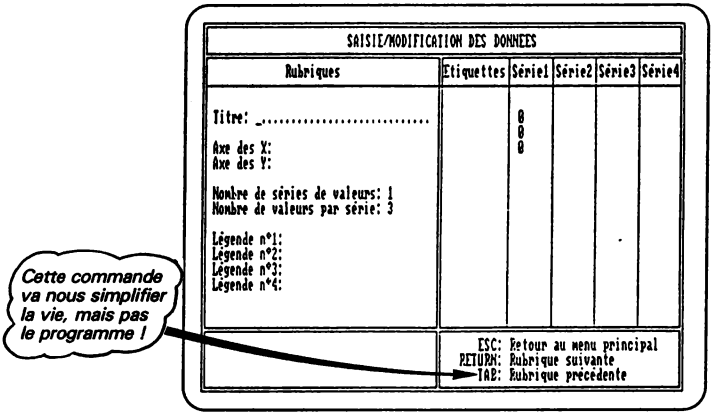


Fig. 4.6. : Ecran d'entrée des données

Ceci étant, commençons par le commencement, c'est-à-dire par le masque d'écran :

```
320 'Saisie/Modification des données-----------------------------------------
330 COLOR 15,1,0:CLS:PRINT TRAIT1$:PRINT TRAIT2$:
    PRINT TRAIT7$:PRINT TRAIT8$:PRINT TRAIT9$
340 FOR I=1 TO 15:PRINT TRAIT8$:NEXT:PRINT TRAIT10$
350 FOR I=1 TO 3:PRINT TRAIT5$;:NEXT:LOCATE 25,1:PRINT TRAIT6$;
360 COLOR 14:LOCATE 2,25:PRINT"SAISIE/MODIFICATION DES DONNEES"
370 COLOR 13:LOCATE 4,15:PRINT"Rubriques":LOCATE 4,41:PRINT"Etiquettes"
380 LOCATE 4,52:PRINT"Série1":LOCATE 4,59:PRINT"Série2"
390 LOCATE 4,66:PRINT"Série3":LOCATE 4,73:PRINT"Série4"
400 COLOR 10:LOCATE 22,47:PRINT"ESC: Retour au menu principal"
410 LOCATE 23,44:PRINT"RETURN: Rubrique suivante"
420 LOCATE 24,47:PRINT"TAB: Rubrique précédente";
430 COLOR 11:LOCATE 7,3:PRINT"Titre:"
440 LOCATE 9,3:PRINT"Axe des X:"
450 LOCATE 10,3:PRINT"Axe des Y:"
460 LOCATE 12,3:PRINT"Nombre de séries de valeurs:"
470 LOCATE 13,3:PRINT"Nombre de valeurs par série:"
480 LOCATE 15,3:PRINT"Légende n°1:"
490 LOCATE 16,3:PRINT"Légende n°2:"
500 LOCATE 17,3:PRINT"Légende n°3:"
510 LOCATE 18,3:PRINT"Légende n°4:"
```

Masque de l'écran d'entrée des données

Que peut-on dire sur ce masque d'entrée ?... Rien de bien particulier si ce n'est qu'il est nettement plus chargé que celui du menu principal tout en demeurant relativement bien lisible et ce, grâce à l'emploi des chaînes TRAITS.

Plus intéressante est la suite du programme, car c'est d'elle que dépend l'affichage du contenu des différentes rubriques. Cela va du titre du diagramme aux valeurs des données numériques dont la présence sur l'écran est indispensable dans le cas où l'utilisateur souhaite y apporter quelques modifications.

Affichage du contenu des rubriques

```
520 'Affichage-----------------------------------------------------------------
530 COLOR 15:LOCATE 7,10:PRINT TITRE$
540 FOR I=1 TO 2:LOCATE I+8,14:PRINT AXE$(I):NEXT
550 LOCATE 12,31:PRINT SERIES:LOCATE 13,31:PRINT VALEURS
560 FOR I=1 TO SERIES:LOCATE I+14,16:PRINT LEGENDE$(I):NEXT
570 FOR I=1 TO VALEURS:LOCATE I+6,42:PRINT ETIQUETTE$(I):NEXT
580 FOR J=1 TO SERIES:FOR I=1 TO VALEURS:
    LOCATE I+6,J*7+45:PRINT DONNEE(I,J):NEXT I,J
```

Vous noterez l'emploi de bouches FOR... TO destinées à simplifier notoirement l'écriture de cette portion du programme dans la mesure où certaines rubriques présentent des similitudes (non fortuites...) et aussi parce que l'alignement des séries de valeurs en 4 colonnes (1 colonne par série) le permet.

Ce point étant acquis, il est temps de passer au morceau de bravoure du programme, à savoir l'acquisition des données. Celle-ci devant pouvoir s'effectuer en avant ou en arrière, il a été prévu un numéro d'ordre différent pour chacune des rubriques. Vous trouverez ci-après la liste des numéros d'ordre et des rubriques qui leur correspondent :

N° d'ordre	Rubrique
1	Titre du diagramme
2 et 3	Axes X et Y
4	Nombre de séries
5	Nombre de valeurs/série
6 à 9	Légendes des séries
13 à 24	Etiquettes des séries
25 à 36	Série n° 1 de valeurs
37 à 48	Série n° 2 de valeurs
49 à 60	Série n° 3 de valeurs
61 à 72	Série n° 4 de valeurs

Remarques : ***Les numéros d'ordre 10 à 12 sont libres au cas où vous souhaiteriez introduire de nouvelles rubriques dans cet écran d'entrée (ce n'est pas la place qui manque...).***

```
590 'Acquisition-----------------------------------------------------------
600 RUBRIQUE=1
610 IF RUBRIQUE=1 THEN Y=7:X=10:MAX=29:ENTREE$=TITRE$:
    GOSUB 1570:TITRE$=ENTREE$:LOCATE Y,X:PRINT TITRE$
620 IF RUBRIQUE=2 OR RUBRIQUE=3 THEN I=RUBRIQUE-1:Y=I+8:X=14:MAX=12:
    ENTREE$=AXE$(I):GOSUB 1570:AXE$(I)=ENTREE$:LOCATE Y,X:PRINT AXE$(I)
630 IF RUBRIQUE=4 THEN Y=12:X=31:MAX=2:ENTREE$=STR$(SERIES):GOSUB 1570:
    SERIES=VAL(ENTREE$):LOCATE Y,X:PRINT SERIES:GOSUB 1360
640 IF RUBRIQUE=5 THEN Y=13:X=31:MAX=3:ENTREE$=STR$(VALEURS):GOSUB 1570:
    VALEURS=VAL(ENTREE$):LOCATE Y,X:PRINT VALEURS:GOSUB 1420
650 IF RUBRIQUE>=6 AND RUBRIQUE<=9 THEN I=RUBRIQUE-5:
    Y=I+14:X=16:MAX=12:ENTREE$=LEGENDE$(I):GOSUB 1570:
    LEGENDE$(I)=ENTREE$:LOCATE Y,X:PRINT LEGENDE$(I)
660 IF RUBRIQUE>=13 AND RUBRIQUE<=24 THEN I=RUBRIQUE-12:
    Y=I+6:X=42:MAX=8:ENTREE$=ETIQUETTE$(I):GOSUB 1570:
    ETIQUETTE$(I)=ENTREE$:LOCATE Y,X:PRINT ETIQUETTE$(I)
670 IF RUBRIQUE>=25 AND RUBRIQUE<=72 THEN J=(RUBRIQUE-13)\12:
    I=RUBRIQUE-(J+1)*12:Y=I+6:X=J*7+45:MAX=5:ENTREE$=STR$(DONNEE(I,J)):
    GOSUB 1570:DONNEE(I,J)=VAL(ENTREE$):LOCATE Y,X:PRINT DONNEE(I,J)
680 IF ERREUR=0 THEN ON TABULATION GOSUB 710,770
690 GOTO 610
```

Ceci effectue l'acquisition des données

Du point de vue fonctionnement, cette portion de programme peut vous sembler bien obscure. En effet, hormis le fait que chaque rubrique ou série de rubriques est appelée à travers un IF... THEN, il n'y a pas une seule instruction INPUT pour entrer les données. Avouez tout de même que ce n'est pas banal !

La raison à cet état de choses réside dans le sous-programme 1570 qui n'est rien d'autre que l'équivalent d'une instruction **INPUT** plus puissante que celle du Basic Microsoft puisqu'elle assure la limitation en longueur des entrées ainsi que l'affichage par défaut de l'ancienne chaîne. C'est dire son intérêt !

En attendant d'analyser dans le détail ce sous-programme d'**INPUT** contrôlé, finissons-en avec les rubriques et plus précisément avec le mécanisme d'incrémentation ou de décrémentation de la variable **RUBRIQUE** propre à assurer la progression en avant ou en arrière au sein des données.

La clé du mystère tiens en une seule ligne, la 680 !... En effet, selon la valeur de la variable **TABULATION** qui peut valoir 1 ou 2 selon que vous appuyez sur la touche [RETURN] ou [TAB] (cela se passe au sein du sous-programme d'INPUT contrôlé...), cette ligne aiguille vers les sous-programmes 710 (d'incrémentation) ou 770 (de décrémentation) de la variable **RUBRIQUE**, à condition bien entendu qu'il n'y ait pas eu d'erreur :

```
700 'Incrémentation----------------------------------------------------
710 RUBRIQUE=RUBRIQUE+1
720 IF RUBRIQUE=6+SERIES THEN RUBRIQUE=RUBRIQUE+7-SERIES
730 DIVIDANDE=RUBRIQUE-VALEURS:IF DIVIDANDE>12 AND
    DIVIDANDE MOD 12=1 THEN RUBRIQUE=DIVIDANDE+12
740 IF RUBRIQUE>(SERIES+2)*12 THEN RUBRIQUE=1
750 RETURN


760 'Décrémentation----------------------------------------------------
770 RUBRIQUE=RUBRIQUE-1
780 IF RUBRIQUE=12 THEN RUBRIQUE=RUBRIQUE-7+SERIES
790 DIVIDANDE=RUBRIQUE-12:IF DIVIDANDE>=12 AND
    DIVIDANDE MOD 12=0 THEN RUBRIQUE=DIVIDANDE+VALEURS
800 IF RUBRIQUE<1 THEN RUBRIQUE=(SERIES+1)*12+VALEURS
810 RETURN
```

Incrémentation...

... ou décrémentation du numéro d'ordre des rubriques

Vous noterez la « légère » complexité de ces deux sous-programmes qui doivent assurer des sauts de rubriques selon que le nombre de séries ou de valeurs par série est inférieur au maximum. Ainsi, si l'on prend l'exemple d'une incrémentation, il suffit que vous n'ayez que 6 valeurs dans chaque série pour devoir sauter de la rubrique 18 (dernière des 6 étiquettes) à la rubrique 25 (première donnée de la 1ère série), de la 30 à la 37, etc, etc...

> **Pour progresser d'avant en arrière au sein des rubriques d'un masque d'entrée, il suffit d'affecter un numéro d'ordre à chacune des rubriques et de les incrémenter ou décrémenter à l'aide des touches [RETURN] et [TAB].**

SOUS-PROGRAMME D'INPUT CONTRÔLÉ

Voyons un peu maintenant cette petite merveille !... Il est vrai qu'elle serait encore mieux si elle était écrite en assembleur, mais cela c'est une autre histoire(*) qui sort du cadre de cet ouvrage :

```
1560 'Sous-programme d'INPUT contrôlé----------------------------------------
1570 INDICE=I:DEBUT=X-1:TABULATION=1
1580 LONGUEUR=LEN(ENTREE$):IF LONGUEUR=0 THEN 1600
1590 FOR I=1 TO LONGUEUR:CARACTERE$(I)=MID$(ENTREE$,I,1):
     X=DEBUT+I:LOCATE Y,X:PRINT CARACTERE$(I):NEXT:ENTREE$=""
1600 FOR I=LONGUEUR+1 TO MAX:CARACTERE$(I)=".":
     X=DEBUT+I:LOCATE Y,X:PRINT CARACTERE$(I):NEXT
1610 CARACTERE$(I)=" ":I=LONGUEUR+1:CURSEUR=95
1620 TOUCHE$=INKEY$:ASCII2=0:IF TOUCHE$="" THEN ASCII1=0:GOTO 1640
1630 ASCII1=ASC(TOUCHE$):IF ASCII1=0 THEN ASCII2=ASC(RIGHT$(TOUCHE$,1))
1640 X=DEBUT+I:LOCATE Y,X:PRINT CARACTERE$(I)
1650 IF ASCII1=9 THEN TABULATION=2:GOTO 1880
1660 IF ASCII1=13 THEN 1880
1670 IF ASCII1=27 THEN 150
1680 IF ASCII2=75 AND I>1 THEN I=I-1
1690 IF ASCII2=77 AND I<=MAX THEN I=I+1
1700 IF ASCII2=82 THEN CURSEUR=CURSEUR-73:IF CURSEUR<22 THEN CURSEUR=95
1710 IF ASCII2=83 THEN GOSUB 1830
1720 X=DEBUT+I:LOCATE Y,X:PRINT CHR$(CURSEUR)
1730 IF ASCII1<32 OR ASCII1>126 THEN 1620
1740 IF I>MAX THEN BEEP:GOTO 1620
1750 IF I>=LONGUEUR OR CURSEUR=95 THEN 1790
1760 FIN=LONGUEUR+1:IF FIN>MAX THEN FIN=MAX
1770 FOR I=FIN TO I+1 STEP -1:CARACTERE$(I)=CARACTERE$(I-1):
     X=DEBUT+I:LOCATE Y,X:PRINT CARACTERE$(I):NEXT
1780 IF LONGUEUR<MAX THEN LONGUEUR=LONGUEUR+1
1790 CARACTERE$(I)=TOUCHE$
1800 X=DEBUT+I:LOCATE Y,X:PRINT CARACTERE$(I)
1810 IF I>LONGUEUR THEN LONGUEUR=I
1820 I=I+1:GOTO 1620
1830 IF I>LONGUEUR THEN RETURN
```

Affichage de l'ancienne chaîne ENTREE$ (ligne 1590)

Extinction... (ligne 1640)

... et allumage du curseur clignotant ! (ligne 1720)

Enregistrement du nouveau caractère (ligne 1790)

(*) Lire à ce sujet l'ouvrage « 8088 Assembleur » de H. Lilen paru aux Editions Radio

```
1840 POSITION=I:IF I=LONGUEUR THEN 1860
1850 FOR I=POSITION TO LONGUEUR-1:CARACTERE$(I)=CARACTERE$(I+1):
     X=DEBUT+I:LOCATE Y,X:PRINT CARACTERE$(I):NEXT
1860 CARACTERE$(I)=".":X=DEBUT+I:LOCATE Y,X:PRINT CARACTERE$(I)
1870 LONGUEUR=LONGUEUR-1:I=POSITION:RETURN
1880 IF LONGUEUR=0 THEN 1970
1890 I=1:IF CARACTERE$(I)<>" "THEN 1920
1900 FOR I=1 TO LONGUEUR-1:CARACTERE$(I)=CARACTERE$(I+1):NEXT
1910 LONGUEUR=LONGUEUR-1:GOTO 1890
1920 IF LONGUEUR=0 THEN 1970
1930 FOR I=LONGUEUR TO 2 STEP-1
1940 IF CARACTERE$(I)<>" "THEN 1960
1950 LONGUEUR=LONGUEUR-1:NEXT
1960 FOR I=1 TO LONGUEUR:ENTREE$=ENTREE$+CARACTERE$(I):NEXT
1970 I=INDICE:X=DEBUT+1:LOCATE Y,X:PRINT SPC(MAX)
1980 RETURN
```

Mise en forme de la nouvelle chaîne ENTREE$

Tout d'abord, il faut se rappler que ce sous-programme ne peut fonctionner qu'à condition qu'on fixe la valeur des variables **Y** (ligne de tabulation), **X** (colonne de tabulation) et **MAX** (nombre maximal de caractères que l'on peut taper) préalablement à son appel (voir pour cela les lignes 610 à 670). Par contre, rien n'oblige à transférer le contenu de l'ancienne chaîne dans **ENTREE$**. Cela dépend si l'on souhaite afficher ou non par défaut son précédent contenu.

Ceci étant, le sous-programme débute en affectant la valeur I à la variable **INDICE** (ça, c'est pour retrouver à la sortie du sous-programme la même valeur de I qu'à l'entrée), X-1 à la variable **DEBUT** et 1 à la variable **TABULATION** (c'est-à-dire qu'on incrémente d'office par défaut le numéro d'ordre des rubriques).

Après cela, vous observerez que la chaîne **ENTREE$** (qui contient l'ancienne chaîne) est analysée caractère par caractère (lignes 1580 et 1590) de façon à transférer ceux-ci dans un tableau dénommé **CARACTERE$** (-) avant de les afficher. Ceci fait, on s'empresse d'afficher à leur suite toute une série de petits points (lignes 1600 et 1610) destinés à renseigner l'utilisateur du nombre maximal de caractères dont il dispose. Vous noterez qu'un espace clôture le tout.

Ce point étant acquis, l'indice I est alors incrémenté de façon à afficher le curseur (code ASCII=95) à la suite du dernier caractère de l'ancienne chaîne. Puis c'est au tour du clavier à être interrogé pour déterminer si une touche a été enfoncée (lignes 1620 et 1630) et si oui, laquelle. A ce sujet, il y a lieu de noter qu'il faut parfois aller chercher le 2e octet pour certaines touches spéciales (comme les flèches par exemple). D'ailleurs à ce propos, vous trouverez la liste des codes ASCII étendus en annexe à la fin de cet ouvrage.

Le ou les codes ASCII de la touche enfoncée étant désormais connus, on assiste alors à l'affichage du caractère situé sous le curseur, c'est-à-dire celui qui apparaît lorsque le curseur clignotant est éteint (ligne 1650). Celui-ci d'ailleurs ne tarde pas à se rallumer (ligne 1720) et c'est le bouclage de la ligne 1820 qui assure son clignotement régulier. Remarquez qu'entre le rallumage du curseur et la ligne de bouclage figurent un certain nombre de lignes destinées à traiter le caractère reçu. En effet, ce dernier subit un premier filtrage destiné à s'assurer qu'il s'agit bien d'un caractère alphanumérique (ligne 1730), suivi d'un second qui le rejète impitoyablement si sa position excède la limite autorisée (ligne 1740). Si le caractère arrive à franchir victorieusement ces deux premiers barrages, il saute aussitôt par-dessus les lignes 1760 et 1770 (ces lignes ne servent qu'en cas d'insertion) pour aboutir à la ligne 1790 où il est enfin accepté par **CARACTERE$** (-) et affiché (ligne 1800). La longueur de la chaîne est alors actualisée et la position du curseur incrémentée (lignes 1810 et 1820).

Variable	Désignation
ASCII1	Premier code ASCII d'une touche
ASCII2	Second code ASCII d'une touche
CARACTERES$ (-)	Caractère venant d'être frappé
CURSEUR	Code ASCII du curseur
DEBUT	Début de tabulation horizontale
ENTRÉES$	Chaîne d'entrée des caractères
FIN	Dernier caractère de la chaîne d'entrée
I	Variable à tout faire
INDICE	Mémorise le I extérieur
LONGUEUR	Longueur de la chaîne d'entrée
POSITION	Position d'un caractère dans la chaîne d'entrée
TABULATION	Signale l'enfoncement de la touche [TAB]
TOUCHES$	Chaîne de la touche enfoncée
X	Tabulation horizontale
Y	Tabulation verticale

Sur le plan des touches de commande, vous allez trouver les ordres qui leur correspondent entre les lignes d'extinction et d'allumage du curseur (lignes 1650 à 1710). Ce sont respectivement :

* *la touche* [TAB] *(Code ASCII = 9)* qui aiguille vers la sortie (lignes 1880 à 1980) après avoir affecté la valeur 2 à la variable **TABULATION.**

* *la touche* [RETURN] *(Code ASCII = 13)* qui aiguille également vers la sortie, mais cette fois-ci sans toucher à la valeur initiale (égale à 1) de la variable **TABULATION.**

* *la touche* [ESC] *(Code ASCII = 27)* qui renvoie l'utilisateur vers le menu principal. Ainsi, il est possible de sortir à tout instant de l'écran de saisie des données où que l'on se trouve.

* *la touche* [←] *(Second code ASCII = 75)* qui assure le déplacement du curseur vers la gauche à la condition toutefois que ce dernier ne dépasse pas la limite inférieure.

* *la touche* [→] *(Second code ASCII = 77)* qui déplace le curseur vers la droite. Notez qu'en matière de limite, on autorise cette fois-ci le curseur à aller une position plus loin que le dernier caractère saisissable (de façon à le dégager).

* *la touche* [INS] *(Second code ASCII = 82)* qui autorise l'insertion du caractère saisi dans la chaîne existante, après avoir toutefois changé de curseur pour signaler à l'utilisateur qu'il est en mode d'insertion.

* *la touche* [DEL] *(Second code ASCII = 83)* qui aiguille vers le sous-programme de suppression d'un caractère (lignes 1830 à 1870).

Ceci étant, la fin du sous-programme d'INPUT contrôlé est sans grande surprise. On y rencontre les lignes d'insertion d'un caractère entre les caractères existants (lignes 1760 et 1770) auxquelles on ne peut accéder qu'à la condition d'être en mode d'insertion (code ASCII du curseur clignotant égal à 22) et de ne pas être placé à la suite du dernier caractère. Notez au passage le décalage vers la droite des caractères situés à la droite du curseur.

Dans le même ordre d'idées, vous allez trouver un décalage similaire, mais cette fois-ci vers la gauche (lignes 1840 et 1850), dans le sous-programme de suppression d'un caractère (lignes 1830 à 1870), à la condition toutefois de ne pas être placé à la fin de la chaîne (à cet endroit, il n'y a plus rien à supprimer...).

Enfin, avant de quitter ce sous-programme (c'est le cas de le dire...) vous allez devoir supprimer de la chaîne saisie les éventuels espaces placés à l'avant (lignes 1880 à 1910) ou à l'arrière (lignes 1920 à 1950) de celle-ci de façon à pouvoir remettre en forme la chaîne ENTREES$ (lignes 1960 à 1980) selon les règles de l'art.

Bien au-delà de son utilisation dans l'écran d'entrée des données, vous pouvez employer ce sous-programme d'INPUT contrôlé dans tous vos programmes en lieu et place de l'instruction INPUT habituelle.

ÉCRAN DE SAUVEGARDE ET DE CHARGEMENT DES DONNÉES

Allez, du courage, plus que deux écrans et c'est fini !... Le suivant à entrer en lisse est l'écran de sauvegarde et de chargement des données. Celui-là n'est pas bien compliqué à bâtir !... il suffit que l'utilisateur précise le mode dans lequel il souhaite travailler, c'est-à-dire sauvegarde ou chargement, et qu'après cela il indique le nom du fichier devant être sauvegardé ou chargé :

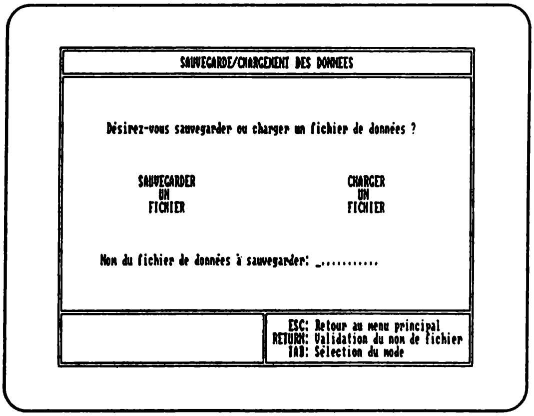


Fig. 4.7. : Ecran de sauvegarde et de chargement des données.

Pour aboutir à un tel résultat, il suffit de bien peu de choses, surtout si l'on profite de la présence du sous-programme d'INPUT contrôlé pour effectuer la saisie du nom du fichier. Mieux encore !... L'on va pouvoir effectuer la sélection du mode grâce à la touche TAB que gère ce sous-programme :

```
820 'Sauvegarde/chargement des données----------------------------------------
830 COLOR 15:CLS:PRINT TRAIT1$:PRINT TRAIT2$:PRINT TRAIT3$:
840 FOR Y=4 TO 20::PRINT TRAIT2$:NEXT:PRINT TRAIT4$
850 FOR Y=22 TO 24:PRINT TRAIT5$;:NEXT:LOCATE 25,1:PRINT TRAIT6$;
860 COLOR 14:LOCATE 2,24:PRINT"SAUVEGARDE/CHARGEMENT DES DONNEES"
870 COLOR 11:LOCATE 7,10:PRINT"Désirez-vous ";
    "sauvegarder ou charger un fichier de données ?"
880 COMMANDE=COMMANDE+1:IF COMMANDE>2 THEN COMMANDE=1
890 IF COMMANDE=1 THEN COLOR 15:LOCATE 11,16:PRINT"SAUVEGARDER":
    LOCATE 12,20:PRINT"UN":LOCATE 13,18:PRINT"FICHIER":COLOR 7:LOCATE 11,56:
    PRINT"CHARGER":LOCATE 12,58:PRINT"UN":LOCATE 13,56:PRINT"FICHIER"
900 IF COMMANDE=2 THEN COLOR 7:LOCATE 11,16:PRINT"SAUVEGARDER":
    LOCATE 12,20:PRINT"UN":LOCATE 13,18:PRINT"FICHIER":COLOR 15:LOCATE 11,56:
    PRINT"CHARGER":LOCATE 12,58:PRINT"UN":LOCATE 13,56:PRINT"FICHIER"
910 COLOR 11:IF COMMANDE=1 THEN LOCATE 17,9:
    PRINT"Nom du fichier de données à sauvegarder:"
920 IF COMMANDE=2 THEN LOCATE 17,9:PRINT SPC(4);
    "Nom du fichier de données à charger:"
930 COLOR 10:LOCATE 22,45:PRINT"ESC: Retour au menu principal"
940 LOCATE 23,42:PRINT"RETURN: Validation du nom de fichier"
950 LOCATE 24,45:PRINT"TAB: Sélection du mode";
960 COLOR 15:Y=17:X=50:MAX=12:ENTREE$=FICHIER$:GOSUB 1570:
    FICHIER$=ENTREE$:LOCATE Y,X:PRINT FICHIER$
970 IF TABULATION=2 THEN 880
980 ON COMMANDE GOSUB 1010,1080
990 GOTO 150
```

Dans cet écran, la touche [TAB] sélectionne le mode de sauvegarde ou de chargement du fichier des données

Comment tout cela fonctionne ?... De façon identique aux autres menus !... Après avoir dessiné le traditionnel cadre autour de l'écran (lignes 830 à 850), le programme affiche le titre de celui-ci (ligne 860) et pose la question de savoir ce que l'utilisateur souhaite faire (ligne 870). Selon la réponse qui en résulte (enfoncement ou non de la touche [TAB]), la variable **COMMANDE** vaut 1 (sauvegarde) ou 2 (chargement) et l'une des lignes 890 ou 900 est exécutée de façon à afficher en clair le mode sélectionné. Il en est de même pour l'une des deux lignes suivantes qui demande que soit précisé le nom du fichier à sauvegarder (ligne 910) ou à charger (ligne 920).

Ceci étant, on retrouve aux lignes 930 à 950 les habituelles indications des touches importantes [ESC], [RETURN] et [TAB], suivies par l'appel du sous-programme d'**INPUT** contrôlé (ligne 970) ainsi que par l'ordre de branchement calculé vers les sous-programmes de sauvegarde ou de chargement :

```
1000 'Sauvegarde-----------------------------------------------------------
1010 OPEN FICHIER$ FOR OUTPUT AS#1
1020 WRITE#1,TITRE$,AXE$(1),AXE$(2),SERIES,VALEURS
1030 FOR I=1 TO SERIES:WRITE#1,LEGENDE$(I):NEXT
1040 FOR I=1 TO VALEURS:WRITE#1,ETIQUETTE$(I):NEXT
1050 FOR J=1 TO SERIES:FOR I=1 TO VALEURS:
     WRITE#1,DONNEE(I,J):NEXT I,J:WRITE#1,DIAGRAMME
1060 CLOSE#1:RETURN
1070 'Chargement-----------------------------------------------------------
1080 OPEN FICHIER$ FOR INPUT AS#1
1090 INPUT#1,TITRE$,AXE$(1),AXE$(2),SERIES,VALEURS
1100 FOR I=1 TO SERIES:INPUT#1,LEGENDE$(I):NEXT
1110 FOR I=1 TO VALEURS:INPUT#1,ETIQUETTE$(I):NEXT
1120 FOR J=1 TO SERIES:FOR I=1 TO VALEURS:
     INPUT#1,DONNEE(I,J):NEXT I,J:INPUT#1,DIAGRAMME
1130 CLOSE#1:RETURN
```

Sauvegarde...

... ou chargement du fichier de données

Vous reconnaissez au passage (*) les habituelles instructions d'écriture et de lecture dans un fichier séquentiel. Notez que les données y sont inscrites (ou lues) dans l'ordre qui est le leur sur l'écran de saisie, en débutant par le titre du diagramme et en terminant par la ou les séries de valeurs.

Plutôt que de sauvegarder des diagrammes, il est bien plus judicieux de ne sauvegarder que les données de ce diagramme, ceux-ci étant redessinés par le programme.

ÉCRAN DE SÉLECTION D'UN DIAGRAMME

Le dernier des écrans à créer est celui qui assure la sélection d'un diagramme parmi la liste suivante (que vous pouvez modifier à votre gré, bien entendu) :

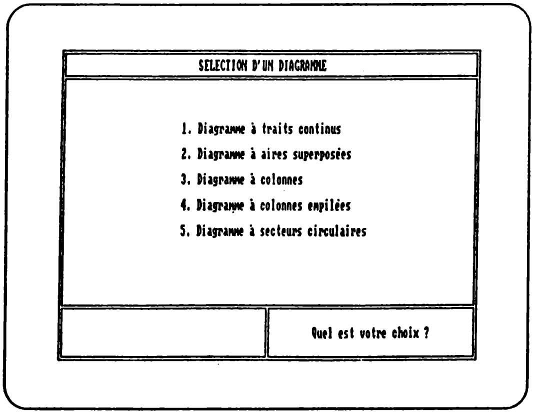


Fig. 4.8. : Ecran de sélection d'un diagramme

Autant dire qu'avec la batterie de sous-programmes déjà existants, l'écriture de la gestion d'un tel écran va se résumer à ces quelques lignes que voici :

(*) Si ce n'est pas le cas, vous pouvez consulter avec profit le volume 2 de Pratique des IBM-PC et compatibles.

```
1140 'Sélection d'un diagramme-----------------------------------------
1150 COLOR 15,1:CLS:PRINT TRAIT1$:PRINT TRAIT2$:PRINT TRAIT3$
1160 FOR I=1  TO 17:PRINT TRAIT2$:NEXT:PRINT TRAIT4$
1170 FOR I=1 TO 3:PRINT TRAIT5$;:NEXT:LOCATE 25,1:PRINT TRAIT6$;
1180 COLOR 14:LOCATE 2,27:PRINT"SELECTION D'UN DIAGRAMME"
1190 COLOR 11:LOCATE 7,24:PRINT"1. Diagramme à traits continus"
1200 LOCATE 9,24:PRINT"2. Diagramme à aires superposées"
1210 LOCATE 11,24:PRINT"3. Diagramme à colonnes"
1220 LOCATE 13,24:PRINT"4. Diagramme à colonnes empilées"
1230 LOCATE 15,24:PRINT"5. Diagramme à secteurs circulaires"
1240 COLOR 10:LOCATE 23,49:PRINT"Quel est votre choix ?"
1250 DIAGRAMME$=INPUT$(1):DIAGRAMME=VAL(DIAGRAMME$):
     COLOR 15:LOCATE 23,71:PRINT DIAGRAMME
1260 GOSUB 1470:IF ERREUR<>0 THEN 1250
1270 ON DIAGRAMME GOTO 1990,1990,1990,1990,1990
1280 GOTO 150
```

C'est ici que se fait la sélection du diagramme (→ ligne 1250)

La seule petite remarque que l'on peut y faire concerne la ligne 1270 dans laquelle on appelle le sous-programme 1480. Or, que fait ce sous-programme ?... Il traite les messages d'erreurs qui peuvent résulter de la frappe de touches non autorisées. Un point c'est tout !

> **L'emploi de sous-programmes bien définis permet de raccourcir notablement l'écriture des programmes comportant de nombeux écrans de menu.**

Solution des exercices sur le chapitre 2

1. Ni l'un, ni l'autre. La réponse est Oui, à condition d'employer l'option B (pour BOX).
2. Bien que cela soit inutile car celle-ci est fixée par défaut, vous pouvez si le cœur vous en dit rajouter la valeur du facteur de correction à la suite de l'instruction CIRCLE (5/6 en moyenne résolution et 5/12 en haute résolution)
3. Ni l'un, ni l'autre. Cela correspond au rapport de l'axe vertical sur l'axe horizontal de l'ellipse, multiplié par la valeur 5/6 (moyenne résolution) ou 5/12 (haute résolution).
4. En employant des valeurs négatives de début et de fin d'arc.
5. Non, elle ne le peut pas.
6. Lorsque le nombre de paires de branches est impair.

Exercices sur le chapitre 4

1. A quoi cela sert-il d'employer de nombreux sous-programmes au sein d'un programme ?

   - ☐ A en simplifier l'écriture.
   - ☐ A en accélérer l'exécution.
   - ☐ A rien, si ce n'est à le rendre plus lisible.

2. Faut-il appuyer sur [RETURN] lorsqu'on emploie un INPUT$ (1) ?

   - ☐ Oui, si l'on veut que le caractère frappé soit enregistré
   - ☐ Non, car cette instruction en dispense.

3. Pour progresser d'avant en arrière au sein d'un groupe de rubriques, il faut :

   - ☐ Que celles-ci soient alignées sur l'écran
   - ☐ Que chaque rubrique possède un numéro d'ordre.

4. Que permet le sous-programme d'INPUT contrôlé que ne permet pas l'instruction INPUT ?

   - ☐ Le contrôle de la longueur des entrées
   - ☐ L'affichage du précédent contenu de la chaîne
   - ☐ la suppression des espaces placés avant et après la chaîne
   - ☐ La gestion permanente des touches [ESC] et [TAB]

5. Pourquoi est-il plus intéressant de sauvegarder des données graphiques plutôt que des diagrammes ?

   - ☐ Parce que cela prend moins de place sur la disquette
   - ☐ Parce que cela permet de les modifier aussi souvent qu'on le désire.

Solution des exercices sur le chapitre 3.

1. Oui, à condition d'employer l'option F.
2. Ni l'un, ni l'autre !... C'est uniquement lorsque la couleur du pourtour de cette instruction est différente de celle du pourtour de la figure.
3. Oui, à condition d'alterner les points rouges et jaunes.
4. Oui, à condition d'employer un fond bleu et de n'allumer en rouge qu'un point sur deux.
5. La couleur rouge (ou violette) avec une densité de points de 3/4.
6. Parce que vous disposez de deux fois plus de points pour les coder.

CHAPITRE V

LES DIAGRAMMES DE GESTION

Le présent chapitre, complément du précédent, a pour vocation de vous hisser à la maîtrise des diagrammes de gestion en vous guidant à travers la programmation de cinq d'entre-eux. Choisi parmi les plus employés, ils vous serviront de base chaque fois que vous chercherez à élaborer celui que vous ne trouvez nulle part.

Menu du chapitre 5

Les diagrammes à traits continus
Les diagrammes à aires superposées
Les diagrammes à colonnes
Les diagrammes à colonnes empilées
Les diagrammes à secteurs circulaires.

Quel est votre choix : ■

LES DIAGRAMMES À TRAITS CONTINUS

Tout d'abord, qu'est-ce qu'un diagramme à traits continus ?... C'est le plus simple de tous les diagrammes, car il n'est constitué que par une succession de traits reliant entre-eux les points représentatifs de l'amplitude des données. D'ailleurs plutôt qu'un long discours, voici un exemple de diagramme à traits continus :

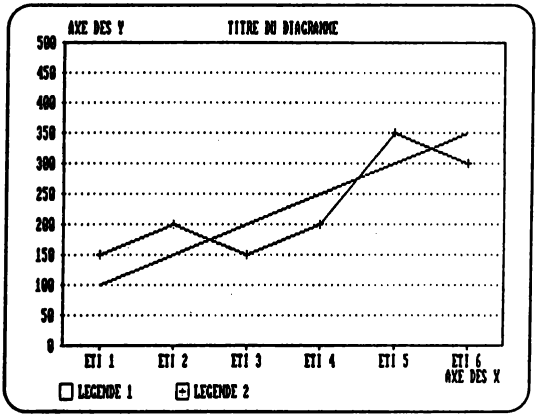


Fig. 5.1. : Diagramme à traits continus

Comme vous pouvez le constater « de visu », un tel diagramme ne s'obtient pas seulement en reliant entre-eux par des traits un certain nombre de points. Il lui faut un repère orthogonal délimité par deux axes, l'un (l'axe des X) étant porteur d'étiquettes (ce peuvent être des jours, des mois, des années, etc, etc...), l'autre (l'axe des Y) étant porteur de graduations équidistantes. De surcroît, le calcul de l'échelle de l'axe des Y se doit d'être automatique de façon à décharger l'utilisateur de cette servitude. En l'occurence, la valeur maximale de celle-ci est alignée sur 500 de façon à s'ajuster au mieux à la valeur la plus élevée des données qui, dans cet exemple, est de 300.

Mais ce n'est pas tout !... Il faut encore rajouter les pointillés horizontaux sans lesquels la lecture du diagramme perdrait en précision, les unités de l'axe des Y et éventuellement celles de l'axe des X lorsque le besoin s'en fait sentir, le titre du programme et pour finir la légende associée à chaque courbe.

Que de travail en perspective !... Commençons donc par le début, c'est-à-dire par le dessin du repère orthogonal, les graduations des deux axes et les pointillés horizontaux. Et comme on va faire appel à ce petit bout de programme pour quatre des cinq diagrammes, on va le loger dans le sous-programme que voici :

```
2970 'Dessin des axes gradués------------------------------------------
2980 LINE(58,11)-(634,171),,B:LINE(59,11)-(635,171),,B
2990 ESPACEMENT=576/VALEURS
3000 FOR I=1 TO VALEURS:X=58+(I-.5)*ESPACEMENT:
     LINE(X,171)-(X,173):NEXT
3010 FOR Y=11 TO 171 STEP 16:LINE(56,Y)-(61,Y):
     LINE(64,Y)-(631,Y),,,&H404:NEXT
3020 RETURN
```

On ne peut pas dire qu'il soit long !... En l'examinant de plus près, vous reconnaîtrez à la ligne 2980 le dessin d'un cadre rectangulaire de 576 × 160 points dont les traits verticaux sont doublés (on est en haute résolution) de façon à paraître aussi épais que les traits horizontaux. Puis, après calcul de l'espacement des graduations sur l'axe des X (ligne 2990), celles-ci sont dessinées uniquement en-dessous de cet axe (vous comprendrez le pourquoi de ce point de détail lorsque nous aborderons l'étude des diagrammes à colonnes).

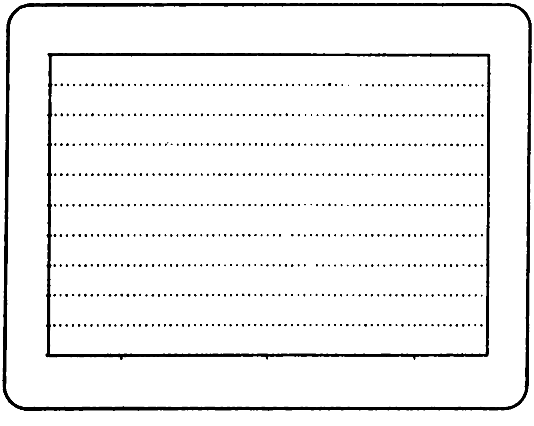


Fig. 5.2. Dessin du repère orthogonal.

Ceci étant, c'est au tour de la ligne 3010 de dessiner les graduations ainsi que les traits pointillés qui les prolongent. Vous noterez le choix de la valeur hexadécimale 0404 (le premier zéro n'apparaît pas sur le listage) comme code d'allumage des points. Pourquoi une telle valeur ?... Pour qu'il n'y ait pas plus d'un point d'allumé tous les 8 points et que ceux-ci soient alignés sur les graduations de l'axe des X (vous saisirez toute l'importance de la « chose » lorsque nous en viendrons à colorier les diagrammes).

Ce premier point étant acquis, passons maintenant au calcul automatique de l'échelle de l'axe des Y. Sur quel principe est-il basé ?... Sur la division ou la multiplication par 10 de la donnée ayant la valeur la plus élevée (il s'agit de l'extrémum de la fonction) jusqu'à ce que le résultat soit compris entre 1.000000 et 9.999999, en notant au passage le nombre de divisions ou de multiplications successives qu'il a fallu accomplir pour aboutir à ce résultat. On obtient alors un cœfficient multiplicateur (ou diviseur selon le cas) qui est une puissance de dix et dont le nombre de zéros correspond au nombre de multiplications ou de divisions successives qu'il a fallu faire. Ainsi, la valeur maximale 1800 donnera lieu à 3 divisions successives par 10 pour aboutir à une mantisse de 1.8 et à un cœfficient multiplicateur de 1000.

En possession de ces deux valeurs (mantisse et cœfficient multiplicateur), le reste du calcul automatique de l'échelle devient tout simple. Il suffit pour cela de comparer la mantisse à toute une série de valeurs de fin d'échelle et à attribuer à celle-ci la valeur de fin d'échelle qui lui est la plus proche, non sans oublier de multiplier (ou diviser) cette nouvelle mantisse par le cœfficient multiplicateur. C'est d'ailleurs ce que fait le sous-programme suivant :

```
3030 'Calcul de l'échelle de l'axe des Y-----------------------------------
3040 MULTIPLICATEUR=1
3050 IF EXTREMUM=0 THEN RETURN
3060 IF EXTREMUM<=1 THEN EXTREMUM=EXTREMUM*10:
     MULTIPLICATEUR=MULTIPLICATEUR/10:GOTO 3060
3070 IF EXTREMUM>10 THEN EXTREMUM=EXTREMUM/10:
     MULTIPLICATEUR=MULTIPLICATEUR*10:GOTO 3070
3080 IF EXTREMUM<=2 THEN EXTREMUM=2:GOTO 3110
3090 IF EXTREMUM<=5 THEN EXTREMUM=5:GOTO 3110
3100 IF EXTREMUM<=10 THEN EXTREMUM=10
3110 EXTREMUM=EXTREMUM*MULTIPLICATEUR
3120 ECHELLE=160/EXTREMUM
3130 RETURN
```

Le cœfficient **MULTIPLICATEUR** étant fixé à 1 au départ (ligne 3040), l'**EXTREMUM** des données est alors multiplié ou divisé par 10 selon qu'il est inférieur ou égal à 1 (ligne 3060) ou supérieur à 10 (ligne 3070). Vous noterez que c'est au sein de chacune de ces deux lignes que le cœfficient **MULTIPLICATEUR** acquiert sa valeur.

A la sortie de ces deux lignes, on aboutit aux comparaisons de l'**EXTREMUM** mantisse avec les valeurs 2, 5 et 10 de fin d'échelle (lignes 3080 à 3100). Vous noterez que ces comparaisons s'effectuent dans l'ordre croissant et qu'il est impossible d'attribuer plusieurs valeurs successives à la mantisse grâce aux **GOTOs** 3110 qui renvoient sur la ligne de calcul de la valeur complète de fin d'échelle. La ligne 3120, quant à elle, détermine grâce à cette valeur l'**ECHELLE** de l'axe de Y.

Ce deuxième point étant acquis, il reste à aborder le sous-programme d'affichage des inscriptions. Que peut-on en dire ?... Tout d'abord, qu'il doit afficher le titre du diagramme en le centrant sur la ligne du haut. Ensuite, il faut qu'il en fasse de même avec les étiquettes de l'axe des X qui doivent être centrées et réparties le long de cet axe. Dans le même ordre d'idées, il faut que les graduations de l'échelle de l'axe des Y soient justifiées à droite le long de l'axe. Enfin, il faut qu'il sache afficher les légendes à la place qui est la leur (c'est-à-dire sur la 25e ligne), à condition, toutefois que les données correspondantes ne soient pas toutes nulles (sinon, à quoi cela servirait-il d'afficher une légende ?).

Or, tout ceci est réuni dans le petit sous-programme que voici :

```
3140 'Affichage des inscriptions--------------------------------------------
3150 LOCATE 1,44-LEN(TITRE$)/2:PRINT TITRE$
3160 LOCATE 24,80-LEN(AXE$(1)):PRINT AXE$(1);:LOCATE 1,9:PRINT AXE$(2)
3170 FOR I=1 TO VALEURS:ETIQUETTE$=LEFT$(ETIQUETTE$(I),ESPACEMENT/8-1):
     LOCATE 23,8+(I-.5)*ESPACEMENT/8-LEN(ETIQUETTE$)/2:PRINT ETIQUETTE$:NEXT
3180 IF EXTREMUM<>0 THEN FOR I=0 TO 10:GRADUATION=I*EXTREMUM/10:
     LOCATE 22-I*2,7-LEN(STR$(GRADUATION)):PRINT GRADUATION:NEXT
3190 FOR J=1 TO SERIES:SOMME=0:FOR I=1 TO VALEURS:
     SOMME=SOMME+DONNEE(I,J):NEXT:IF SOMME=0 THEN 3210
3200 X=152*J-88:Y=195:LINE(X-8,Y-4)-(X+8,Y+4),,B:
     LOCATE 25,J*19-8:PRINT LEGENDE$(J);
3210 NEXT:RETURN
```

... qui procure le résultat que voilà :

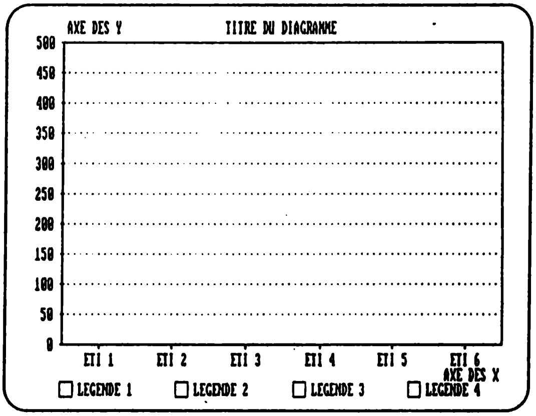


Fig. 5.3. : Affichage des inscriptions

Que peut-on ajouter comme commentaires à ce sous-programme ?

Rien de bien particulier si ce n'est la ligne 3180 qui n'affiche les graduations de l'axe des Y qu'à la condition que l'EXTREMUM des données des séries ne soit pas nul (il peut toutefois être négatif). Egalement, dans le même ordre d'idées, la ligne 3200 n'est exécutée qu'à la condition que la SOMME des données d'une série ne soit pas nulle. Cela évite l'affichage « fâcheux » d'une légende pour des données inexistantes !

Le tour des sous-programmes étant fait, il ne reste plus qu'à aborder le programme de dessin des diagrammes à traits continus. Or, que doit faire un tel programme ?... En fait trois fois rien puisque, tout ce qu'on lui demande, c'est de relier par des traits les points représentatifs des données. La preuve en est les quelques lignes qui suivent :

Vers sous-programme de dessin des axes graduées...

```
2000 SCREEN 2:CLS:GOSUB 2980
2010 EXTREMUM=0:FOR J=1 TO SERIES:FOR I=1 TO VALEURS:
     IF DONNEE(I,J)>EXTREMUM THEN EXTREMUM=DONNEE(I,J)
2020 NEXT I,J:GOSUB 3040:GOSUB 3150
2030 FOR J=1 TO SERIES:SOMME=0:FOR I=1 TO VALEURS:
     SOMME=SOMME+DONNEE(I,J):NEXT:IF SOMME=0 THEN 2110
2040 FOR I=1 TO VALEURS:X=58+(I-.5)*ESPACEMENT
2050 Y=171-DONNEE(I,J)*ECHELLE:IF I=1 THEN PRESET(X,Y)
2060 LINE-(X,Y)
2070 IF J=2 THEN LINE(X,Y-2)-(X,Y+2):LINE(X-4,Y)-(X+4,Y):
     LINE(216,193)-(216,197):LINE(212,195)-(220,195)
2080 IF J=3 THEN LINE(X-2,Y-1)-(X+2,Y+1),,BF:
     LINE(366,194)-(370,196),,BF
2090 IF J=4 THEN LINE(X-4,Y-2)-(X+4,Y+2):LINE(X-4,Y+2)-(Y+4,Y-2):
     LINE(516,193)-(524,197):LINE(516,197)-(524,193)
2100 PRESET(X,Y):NEXT
2110 NEXT
2120 TOUCHE$=INKEY$:IF TOUCHE$="" THEN 2120
2130 IF ASC(TOUCHE$)<>27 THEN 2120
2140 SCREEN 0:GOTO 150
```

... de calcul de l'échelle de l'axe des Y et d'affichage des inscriptions

Retour au menu principal

Vous y reconnaîtrez l'appel du sous-programme de dessin des axes gradués (ligne 2000), suivi par la ligne 2010 de détermination de l'EXTREMUM des données, suivie à son tour par la ligne 2020 d'appel des deux sous-programmes de calcul automatique de l'échelle des Y et d'affichage des inscriptions.

Cela étant, on trouve aux lignes 2040 et 2050 le calcul des coordonnées de chaque point, suivies à la ligne 2060 par le dessin du trait (Eh oui, c'est tout !...). Le reste, c'est-à-dire les lignes 2070 à 2090 ne sert qu'à dessiner une croix (ligne 2070), un petit carré (ligne 2080) ou un grand X à l'emplacement des points des séries 2, 3 et 4, dessin que l'on retrouve également à l'intérieur du cadre qui précède les légendes. Quant à la ligne 2030, elle sert à court-circuiter tout cela lorsqu'une série ne comporte que des valeurs nulles.

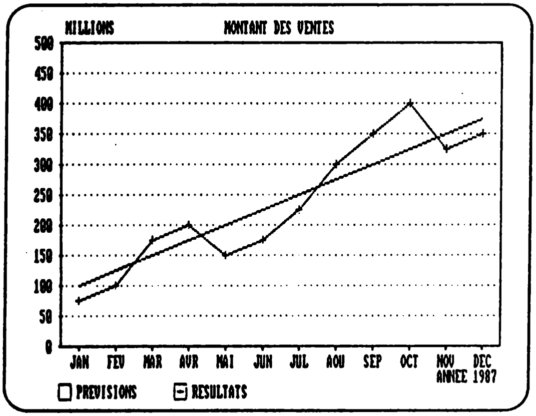


Fig. 5.4. : Exemple d'emploi d'un diagramme à traits continus

A quoi servent les lignes 2120 à 2140 ?... Mais à revenir au menu principal en appuyant (uniquement) sur la touche ESC, un point c'est tout !

Pour obtenir un diagramme à traits continus, il faut :

— définir un repère orthogonal
— calculer automatiquement l'échelle de l'axe des Y
— déterminer l'espacement entre valeurs
— afficher les inscriptions et les traits continus.

LES DIAGRAMMES À AIRES SUPERPOSÉES

Bien que pouvant afficher simultanément 4 séries de valeurs, les diagrammes à traits continus sont difficilement exploitables lorsque les données d'une série ont des valeurs proches de celles d'une autre série. Les traits ont alors tendance à se superposer les uns sur les autres, comme dans l'exemple suivant :

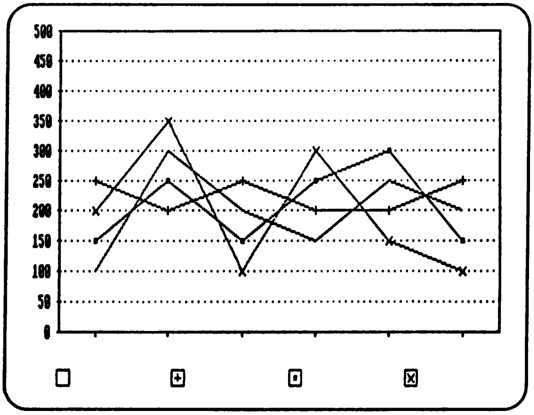


Fig. 5.5. : Données difficilement exploitables avec le diagramme précédent.

La solution à une telle situation consiste à dessiner chaque trait, non plus référencé par rapport à l'axe des X comme c'était le cas précédemment, mais au contraire par rapport au trait précédent. De surcroît, on peut colorier différemment les espaces qui séparent les traits (on parle dans ce cas-là d'aires) de façon à améliorer encore un peu plus la lisibilité de l'ensemble. On obtient alors le diagramme à aires superposées que voici :

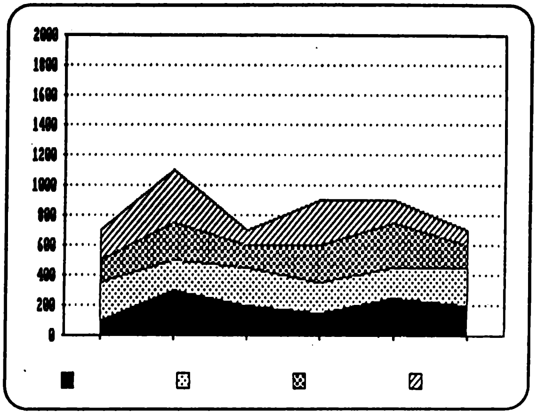


Fig. 5.6. : Les mêmes données présentées cette fois-ci avec un diagramme à aires superposées.

Nettement mieux, n'est-ce pas ?... Et ce n'est pas compliqué à obtenir !... Il suffit de réemployer les sous-programmes précédents, d'y rajouter un sous-programme de coloriage et d'écrire quelques lignes de programme pour y parvenir. Si, donc, cela vous tente, suivez le guide !... Nous allons commencer par le sous-programme de coloriage :

```
3220 'Coloriage des aires et colonnes----------------------------------------
3230 MOTIF$(1)=CHR$(255)+CHR$(255)+CHR$(255)+CHR$(255)
3240 MOTIF$(2)=CHR$(0)+CHR$(3)+CHR$(0)+CHR$(48)
3250 MOTIF$(3)=CHR$(15)+CHR$(15)+CHR$(240)+CHR$(240)
3260 MOTIF$(4)=CHR$(192)+CHR$(3)+CHR$(12)+CHR$(48)
3270 PAINT(X,Y),MOTIF$(J)
3280 PAINT (J*152-88,195),MOTIF$(J)
3290 RETURN
```

Comme vous pouvez le constater, on définit 4 MOTIFs différents de coloriage (un par série de valeurs) aux lignes 3230 à 3260. Ceci fait, on colore chaque aire à l'aide d'un des motifs précédents en débutant le coloriage en un point de coordonnées X, Y (ligne 3270). Il importe donc que celles-ci soient clairement définies par le programme principal. Quant à la ligne 3280, elle dessine à l'intérieur du cadre de chaque légende le même MOTIF que celui de l'aire qui lui correspond (heureusement, d'ailleurs !).

Remarque : ***Pour des raisons toutes personnelles, l'auteur a sélectionné 4 motifs différents qu'il a placé dans l'ordre de la figure 5.6. Bien entendu, ceci n'est pas rédhibitoire et toute initiative personnelle visant à modifier cela ne peut être que bienvenue.***

Cela étant, il nous faut maintenant aborder le programme de dessin des diagrammes à aires superposées. Or, que doit accomplir un tel programme ?... Les mêmes fonctions que précédemment, à l'exception du calcul des coordonnées Y de chaque point qui doit être référencée cette fois-ci par rapport à sa valeur dans la série précédente et non plus par rapport au zéro de l'axe des Y :

Calcul de l'extremum cumulé des valeurs

```
2150 'Diagramme à aires superposées--------------------------------------------
2160 SCREEN 2:CLS:GOSUB 2980
2170 FOR I=1 TO VALEURS:SOMME(I)=0:NEXT
2180 EXTREMUM=0:FOR I=1 TO VALEURS:SOMME=0
2190 FOR J=1 TO SERIES:SOMME=SOMME+DONNEE(I,J)
2200 IF SOMME>EXTREMUM THEN EXTREMUM=SOMME
2210 NEXT J,I:GOSUB 3040:GOSUB 3150
2220 FOR I=1 TO VALEURS:SOMME(I)=0:NEXT
2230 FOR J=1 TO SERIES:FOR I=1 TO VALEURS
2240 X=58+(I-.5)*ESPACEMENT
2250 SOMME(I)=SOMME(I)+DONNEE(I,J)
2260 Y=171-SOMME(I)*ECHELLE
2270 IF I=1 THEN PRESET(X,Y):LINE(X,171)-(X,Y)
2280 LINE-(X,Y):NEXT:LINE(X,171)-(X,Y):NEXT
2290 FOR I=1 TO VALEURS
2300 X=58+(I-.5)*ESPACEMENT:IF I=1 THEN X=X+1
2310 IF I=VALEURS THEN X=X-1
2320 PALIER=171:FOR J=1 TO SERIES
2330 Y=PALIER-DONNEE(I,J)*ECHELLE
2340 Y=Y+1:IF Y<PALIER THEN GOSUB 3230
2350 PALIER=Y-1:NEXT J,I
2360 TOUCHE$=INKEY$:IF TOUCHE$="" THEN 2360
2370 IF ASC(TOUCHE$)<>27 THEN 2360
2380 SCREEN 0:GOTO 150
```

Vers sous-programme de coloriage des aires

C'est pourquoi vous allez rencontrer à la ligne 2160 le désormais classique appel du sous-programme de dessin du repère X, Y, suivi aux lignes 2170 à 2200 du calcul de l'**EXTREMUM** cumulé des valeurs, suivi à son tour par l'appel des deux sous-programmes de calcul automatique de l'échelle et d'affichage des inscriptions (ligne 2210).

La suite, par contre, est différente du diagramme précédent. Elle débute par le calcul des coordonnées X et Y des points (lignes 2230 à 2260). Notez que, dans le calcul de Y, l'on tient compte de la **SOMME** cumulée des points d'un même alignement vertical et ce, dans le but de les disposer l'un au-dessus de l'autre en additionnant leurs valeurs.

Cela étant, vous allez retrouver la méthode traditionnelle de dessin d'une succession de traits avec fixation du point de départ (ligne 2270) et dessin du trait reliant deux points consécutifs (ligne 2280). Notez qu'un trait vertical est systématiquement dessiné entre le premier point et l'axe des X (ligne 2270) ainsi qu'entre le dernier point et le même axe de façon à fermer l'aire pour qu'elle puisse être coloriée.

Ceci fait, il ne reste plus qu'à effectuer le calcul des coordonnées X et Y du point de début du coloriage de chaque aire. C'est le rôle dévolu aux lignes 2290 à 2350. Pourquoi tant de lignes pour si peu de choses vous demandez-vous ?... Mais à cause des points nuls, tels que celui qui apparaît sur la figure 5.7. :

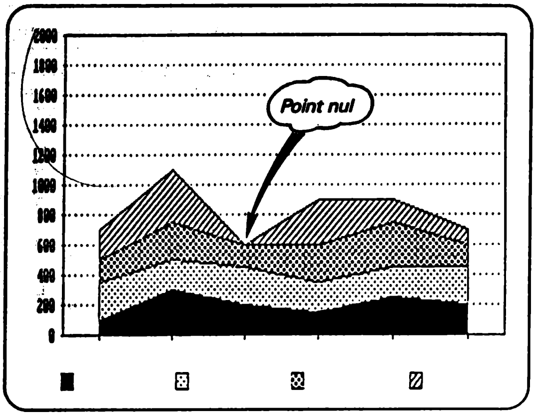


Fig. 5.7. : Les points nuls empêchent le coloriage de se propager dans la totalité de l'aire.

C'est pourquoi il faut relancer systématiquement l'opération de coloriage chaque fois que la coordonnée Y d'un point est supérieure (en fait c'est l'inverse à cause de la numérotation verticale des lignes) d'un pixel au **PALIER** précédent (ligne 2340). Sans cela, le coloriage ne franchirait pas le point nul.

Remarque : *Cette portion de programme ainsi que le sous-programme de coloriage faisant appel aux variables indicées SOMME (I) et MOTIFS (I), il importe de ne pas oublier de les dimensionner en début de programme :*

```
10 KEY OFF
20 DIM AXE$(2),ETIQUETTE$(12),DONNEE(12,4),CARACTERE$(80),SOMME(12),MOTIF$(12)
```

L'obtention d'un diagramme à aires superposées nécessite un cumul des valeurs des données correspondant à un même alignement vertical de points.

LES DIAGRAMMES À COLONNES

Les diagrammes à colonnes améliorent grandement la vision comparative que l'on peut souhaiter avoir entre les données. Je n'en veux pour preuve que celui de la figure 5.8. qui permet de comparer la consommation électrique d'une maison d'une année sur l'autre et ce, mois par mois :

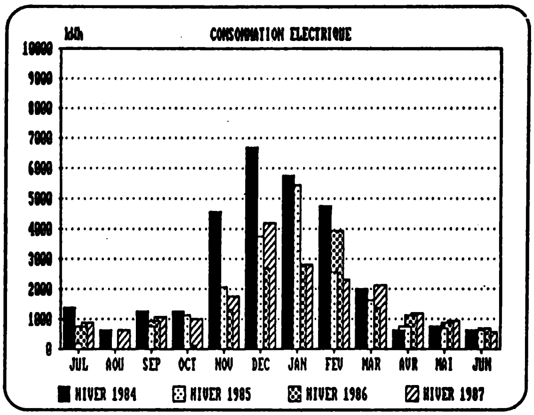
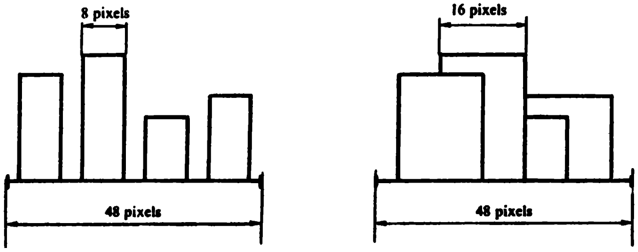


Fig. 5.8. : Un exemple de diagramme à colonnes

Que peut-on extraire comme informations d'un tel diagramme ? Tout d'abord, que la pointe de consommation électrique se situe pendant les mois d'hiver (décembre, janvier ou février). Que celle-ci eut lieu en décembre lors des hivers 1984 et 1987, en janvier lors de l'hiver 1985 et en février lors de l'hiver 1986. Que, de plus, cette pointe fut la plus forte lors de l'hiver 1984 (6660 kwh) et la moins forte lors de l'hiver 1986 (3920 kwh), etc, etc...Comme vous pouvez le voir, un diagramme à colonnes est d'une très grande richesse. Il suffit de l'interroger d'un regard !... Mais avant de pouvoir l'interroger, il faut arriver à présenter les données sous la forme de colonnes et cela, c'est une autre affaire !

Présenter des données sous la forme de colonnes nécessite que soit résolu un problème majeur, à savoir la largeur qu'il faut attribuer à chaque colonne selon le nombre de séries et le nombre de valeurs que l'on veut afficher, compte tenu de l'option de présentation choisie (colonnes de même étiquette placées côte-à-côte ou l'une derrière l'autre).

Un rapide calcul permet de se rendre compte que l'on ne dispose que de 12 pixels par colonne (576/12/4) lorsque l'on cherche à placer les colonnes côte-à-côte. C'est peu, d'autant plus que ce calcul ne tient pas compte de l'intervalle qu'il faut laisser entre colonnes adjacentes et qui doit se monter au minimum à 4 pixels. C'est pourquoi, on lui préfèrera la solution qui consiste à placer les colonnes les unes derrière les autres en les décalant comme si elles étaient vues en perspective.



Comme vous pouvez le constater, cette seconde solution permet de doubler la largeur des colonnes dans un même intervalle de base, ce qui ne peut qu'accroître la lisibilité de l'ensemble. Toutefois, cette disposition un peu inhabituelle se traduit par quelques complications dans le dessin des colonnes selon que la hauteur d'une colonne est plus haute ou plus basse que celle qui est devant elle. Dans un cas (colonne plus haute), il faut dessiner une portion du trait vertical de gauche alors que dans l'autre cas (colonne plus basse), il ne faut dessiner qu'une portion du trait horizontal, le trait vertical quant à lui étant totalement inutile.

C'est donc ces dispositions un peu particulières que vous allez retrouver dans le programme de dessin des diagrammes à colonnes que voici :

```
2390 'Diagramme à colonnes--------------------------------------------
2400 SCREEN 2:CLS:GOSUB 2980
2410 EXTREMUM=0:FOR J=1 TO SERIES:FOR I=1 TO VALEURS:
     IF DONNEE(I,J)>EXTREMUM THEN EXTREMUM=DONNEE(I,J)
2420 NEXT I,J:GOSUB 3040:GOSUB 3150
2430 FOR I=1 TO VALEURS:PALIER=171:FOR J=1 TO SERIES
2440 X=58+(I-.5)*ESPACEMENT+(J-3.5)*96/VALEURS
2450 Y=171-DONNEE(I,J)*ECHELLE
2460 IF Y<PALIER THEN LINE(X,PALIER)-(X,Y)
2470 X=X+(6-SERIES)*96/VALEURS
2480 IF Y<PALIER THEN LINE-(X,Y)
2490 LINE (X-96/VALEURS,Y)-(X,Y):LINE-(X,171)
2500 Y=Y+1:IF Y<171 THEN X=X-1:GOSUB 3230
2510 PALIER=Y-1:NEXT J,I
2520 TOUCHE$=INKEY$:IF TOUCHE$="" THEN 2520
2530 IF ASC(TOUCHE$)<>27 THEN 2520
2540 SCREEN 0:GOTO 150
```

Le dessin d'une colonne est différent selon qu'elle est plus haute ou plus basse que celle qui la précède

Vous allez y retrouver les habituelles lignes de dessin du repère orthogonal (ligne 2400), de calcul de l'**EXTREMUM** des données (ligne 2410), de calcul automatique de l'**ÉCHELLE** et d'affichage des inscriptions (ligne 2420). Après, cela change un peu !...

Cela change au niveau de la ligne 2040 de calcul de la coordonnée X de la ligne vertical gauche de la colonne. Notez qu'elle prend en compte l'**ESPACEMENT** entre valeurs, le nombre de **VALEURS** ainsi que la valeur et la série concernées (variables I et J). Cela change également au niveau de la ligne 2460 qui ne dessine le trait vertical de gauche qu'à la condition que la coordonnée Y soit plus faible (c'est-à-dire corresponde à une colonne plus haute) que le **PALIER** de la colonne précédente. Cela change encore au niveau de la ligne 2480 qui ne dessine la totalité du trait horizontal de la colonne qu'à la condition que celle-ci soit plus haute que le **PALIER** de celle qui la précède.

Pour le reste, tout est pareil. Tout au plus pouvez-vous noter que le point de début du coloriage est placé à 1 pixel à l'intérieur de l'angle supérieur droit de la colonne, ce qui se traduit par X = X - 1 et Y = Y + 1 (ligne 2500). Quant à la ligne 2510, elle attribue la valeur de la coordonnée Y à la variable **PALIER**, le -1 n'étant destiné qu'à compenser le +1 de la ligne précédente.

Dans un diagramme à colonnes, la largeur de chaque colonne dépend du nombre de séries et de valeurs dans chaque série, ainsi que du mode de représentation adapté.

LES DIAGRAMMES À COLONNES EMPILÉES

On pourrait penser que les diagrammes à colonnes représentent le « nec plus ultra » en matière de représentation graphique des données et que, dans ces conditions, il n'y a plus lieu de se creuser la tête à inventer quoi que ce soit d'autre !... Or, est-ce bien vrai ?

La réponse à cette question est bien évidemment non !... Les diagrammes à colonnes permettent de visualiser les données sous un certain angle, un point c'est tout !... Un diagramme peut présenter certains avantages par rapport à un autre sans pour autant être exempt des limitations inhérentes à son mode de représentation. Ainsi, le diagramme à colonnes de l'exemple précédent (figure 5.8.) peut-il nous préciser le mois d'hiver pendant lequel la moyenne de la consommation électrique sur les 4 années fut la plus forte ?... Etait-ce le mois de décembre, janvier ou février ?... Difficile à dire, n'est-ce pas ?

Pourtant, il existe une autre forme de diagramme capable de fournir la réponse à cette question en un clin d'œil. Il s'agit des diagrammes à colonnes empilées !... Or, qu'est-ce qu'un diagramme à colonnes empilées ?... C'est avant tout un diagramme à colonnes, la différence intervenant dans la présentation des colonnes, non plus côte-à-côte ou l'une derrière l'autre, mais juxtaposées l'une au-dessus de l'autre, comme dans l'exemple que voici :

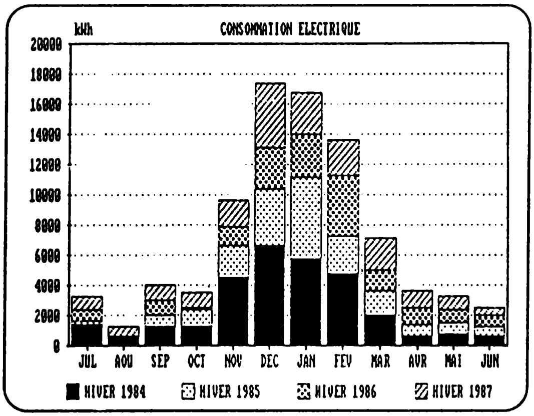


Fig. 5.9. : Un exemple de diagramme à colonnes empilées

Comme vous pouvez le constater par vous-même, la réponse à la question posée est instantanée : il s'agit du mois de décembre !... En fait, les diagrammes à colonne superposées sont par excellence les diagrammes les mieux adaptés à l'analyse moyennée des données, analyse que tout un chacun est appelée à faire un jour ou l'autre et que le programme suivant vous réalise fort simplement :

```
2550 'Diagramme à colonnes empilées----------------------------------------
2560 SCREEN 2:CLS:GOSUB 2980
2570 EXTREMUM=0:FOR I=1 TO VALEURS:SOMME=0
2580 FOR J=1 TO SERIES:SOMME=SOMME+DONNEE(I,J):NEXT
2590 IF SOMME>EXTREMUM THEN EXTREMUM=SOMME
2600 NEXT:GOSUB 3040:GOSUB 3150
2610 FOR I=1 TO VALEURS:LARGEUR=480/VALEURS
2620 PALIER=171:FOR J=1 TO SERIES
2630 X=58+(I-.5)*ESPACEMENT-LARGEUR/2
2640 Y=PALIER-DONNEE(I,J)*ECHELLE
2650 LINE(X,PALIER)-(X,Y)
2660 X=X+LARGEUR:LINE-(X,Y):LINE-(X,PALIER)
2670 Y=Y+1:IF Y<PALIER THEN X=X-1:GOSUB 3230
2680 PALIER=Y-1:NEXT J,I
2690 TOUCHE$=INKEY$:IF TOUCHE$="" THEN 2690
2700 IF ASC(TOUCHE$)<>27 THEN 2690
2710 SCREEN 0:GOTO 150
```

Voyons maintenant un peu comment tout cela fonctionne !... Après sélection du mode graphique à haute résolution et effacement de l'écran (ligne 2560), on retrouve l'appel du sous-programme de dessin du repère orthogonal (même ligne), suivi du calcul de l'**EXTREMUM** cumulé des données (lignes 2570 à 2590), suivi de l'appel des deux sous-programmes de calcul de l'échelle et d'affichage des inscriptions (ligne 2600). Notez au passage que le calcul de l'**EXTREMUM** cumulé s'obtient en faisant la **SOMME** des données d'une même étiquette.

Ceci étant, on procède alors au calcul de la **LARGEUR** des colonnes qui, cette fois-ci, ne dépend que du nombre de valeurs (ligne 2610) et non plus du nombre de séries. Ce qui par contre dépend du nombre de séries, c'est l'empilement des colonnes les unes sur les autres. Notez que, comme dans le diagramme à aires superposées, celui-ci s'obtient en dessinant chaque colonne par rapport à un **PALIER** (ligne 2620), **PALIER** que l'on retrouve dans le calcul de la coordonnée Y de hauteur (ligne 2640).

La suite du programme est sans difficulté !... Les lignes 2650 à 2660 dessinent les deux parois verticales et le palier supérieur de chaque colonne que la ligne 2670 s'évertue à colorier de motifs différents. Notez que comme dans le diagramme précédent, le point de début du coloriage est placé à l'intérieur de chaque colonne, décalé d'un pixel par rapport à l'angle supérieur droit. Notez également que ce coloriage n'est commandé qu'à la condition expresse qu'il existe quelque chose à dessiner. C'est la raison d'être de la ligne 2670 qui teste si la coordonnée Y est plus haute (sur l'écran et non en valeur algébrique). Si ce n'est pas le cas, on s'abstient de lancer le programme de coloriage qui, débutant 1 pixel en dessous, inonderait l'étage inférieur d'un motif qui n'aurait rien à y faire.

Les diagrammes à colonnes empilées s'obtiennent en dessinant les colonnes d'une même étiquette les unes par dessus les autres.

Remarque : *Contrairement au diagramme précédent, vous avez avec celui-ci tout le loisir de modifier à votre gré la LARGEUR des colonnes (lignes 2610). Voici par exemple ce que donne le même diagramme avec des colonnes plus étroites :*

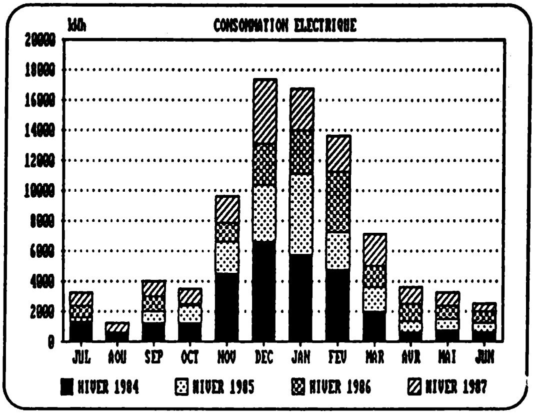


Fig. 5.10 : Diagramme à colonnes empilées plus étroites

LES DIAGRAMMES À SECTEURS CIRCULAIRES

Les diagrammes à secteurs circulaires, encore appelés « camemberts » par les connaisseurs (en fromage bien sûr...), sont le type même de diagrammes parfaitement adaptés à la représentation des données sous forme de pourcentage, la preuve en est celui de la figure 5.11 qui illustre la répartition des ventes de micro-ordinateurs en France selon leur marque :

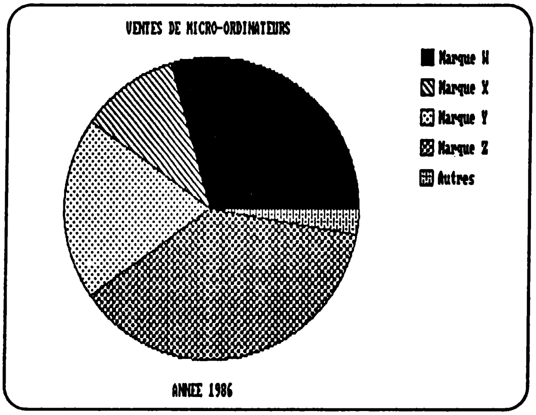


Fig. 5.11 : Répartition des ventes de micro-ordinateurs en France pour l'année 1986.

Or, que constate-t-on à la lecture de ce diagramme ?... Que les micro-ordinateurs de la marque Z l'emportent haut la main !... En un coup d'œil, vous voilà renseigné sur ce point digne d'intérêt grâce à ce diagramme à secteurs circulaires.

Cette constatation étant chose faite, il reste à analyser le programme qui permet d'aboutir à un tel résultat, programme que voici d'ailleurs :

```
2720 'Diagramme à secteurs circulaires------------------------------------
2730 SCREEN 2:CLS:PI=3.141593
2740 LOCATE 1,28-LEN(TITRE$)/2:PRINT TITRE$
2750 FOR J=1 TO SERIES:SOMME=0:FOR I=1 TO VALEURS
2760 SOMME=SOMME+DONNEE(I,J):NEXT:IF SOMME=0 THEN 2930
2770 X0=336-(J MOD 2)*224:Y0=51+INT((J-1)/2)*96:RAYON=96
2780 IF SERIES=1 THEN X0=224:Y0=99:RAYON=192
2790 DEBUT=0:FOR I=1 TO VALEURS:IF DONNEE(I,J)=0 THEN 2890
2800 FIN=DEBUT+DONNEE(I,J)/SOMME*2*PI
2810 CIRCLE(X0,Y0),RAYON,,-DEBUT,-FIN
2820 BISSECTRICE=(DEBUT+FIN)/2
2830 X=X0+RAYON/2*COS(BISSECTRICE)
2840 Y=Y0-RAYON/2*SIN(BISSECTRICE)*5/12
2850 GOSUB 3310
2860 X=504:Y=3+I*16:LINE(X-8,Y-4)-(X+8,Y+4),,B
2870 PAINT (X,Y),MOTIF$(I)
2880 LOCATE 1+I*2,66:PRINT ETIQUETTE$(I);:DEBUT=FIN
2890 NEXT
2900 LOCATE 13+INT((J-1)/2)*12,42-(J MOD 2)*28-LEN(LEGENDE$(J))/2
2910 IF SERIES=1 THEN LOCATE 25,28-LEN(LEGENDE$(J))/2
2920 PRINT LEGENDE$(J);
2930 NEXT
2940 TOUCHE$=INKEY$:IF TOUCHE$="" THEN 2940
2950 IF ASC(TOUCHE$)<>27 THEN 2940
2960 SCREEN 0:GOTO 150
```

Vers sous-programme de coloriage des secteurs circulaires

Pour commencer, celui-ci commence par attribuer à **PI** sa valeur (ligne 2730) avant de passer à la ligne suivante à l'affichage du titre du diagramme (ligne 2740). Vient ensuite l'habituelle ligne de détermination de la **SOMME** des données au sein de chaque série (lignes 2750 et 2760), **SOMME** qui d'ailleurs est mise à profit pour court-circuiter l'ensemble du programme lorsqu'elle est égale à zéro (ligne 2770).

Ceci étant, on passe ensuite à la détermination des coordonnées du centre des cercles (il peut y en avoir un par série) ainsi qu'à celle du rayon des cercles (lignes 2780 et 2790). En ce qui concerne les secteurs circulaires, ce sont les lignes 2800 et 2810 qui se chargent d'en calculer les valeurs d'arc de **DEBUT** et de **FIN** afin que ceux-ci puissent être dessinés par la ligne 2820.

Ce n'est qu'une fois chaque arc tracé que l'on songe à le colorier. Pour cela, on en calcule la **BISSECTRICE** et le demi–**RAYON** de façon à placer le point de début du coloriage en plein milieu du secteur (lignes 2830 à 2850). Notez à ce propos que le sous-programme de coloriage est différent de celui précédemment employé pour les autres diagrammes, car il doit être capable de colorier la totalité des 12 secteurs d'une série complète avec 12 motifs différents :

```
3300 'Coloriage des secteurs circulaires--------------------------------------
3310 MOTIF$(1)=CHR$(255)+CHR$(255)+CHR$(255)+CHR$(255)
3320 MOTIF$(2)=CHR$(192)+CHR$(48)+CHR$(12)+CHR$(3)
3330 MOTIF$(3)=CHR$(192)+CHR$(0)+CHR$(12)+CHR$(0)
3340 MOTIF$(4)=CHR$(240)+CHR$(240)+CHR$(15)+CHR$(15)
3350 MOTIF$(5)=CHR$(136)+CHR$(128)+CHR$(156)+CHR$(128)+
     CHR$(136)+CHR$(8)+CHR$(201)+CHR$(8)
3360 MOTIF$(6)=CHR$(12)+CHR$(51)+CHR$(192)+CHR$(192)+
     CHR$(192)+CHR$(51)+CHR$(12)+CHR$(12)
3370 MOTIF$(7)=CHR$(51)+CHR$(51)+CHR$(51)+CHR$(51)+
     CHR$(51)+CHR$(51)+CHR$(51)+CHR$(51)
3380 MOTIF$(8)=CHR$(3)+CHR$(12)+CHR$(48)+CHR$(192)
3390 MOTIF$(9)=CHR$(1)+CHR$(199)+CHR$(1)+CHR$(0)+
     CHR$(16)+CHR$(124)+CHR$(16)+CHR$(0)
3400 MOTIF$(10)=CHR$(3)+CHR$(15)+CHR$(63)+CHR$(0)
3410 MOTIF$(11)=CHR$(207)+CHR$(48)+CHR$(252)+CHR$(3)
3420 MOTIF$(12)=CHR$(3)+CHR$(3)+CHR$(3)+CHR$(252)
3430 PAINT(X,Y),MOTIF$(I)
3440 RETURN
```

Le coloriage des secteurs est effectué, on passe alors au dessin et au coloriage des cadres des étiquettes (lignes 2870 et 2880), suivis aussitôt par l'affichage des étiquettes correspondantes (lignes 2890), suivi à son tour par l'affichage de la ou des légendes (lignes 2900 à 2920), en dessous de chacun des cercles, comme l'illustre la figure suivante :

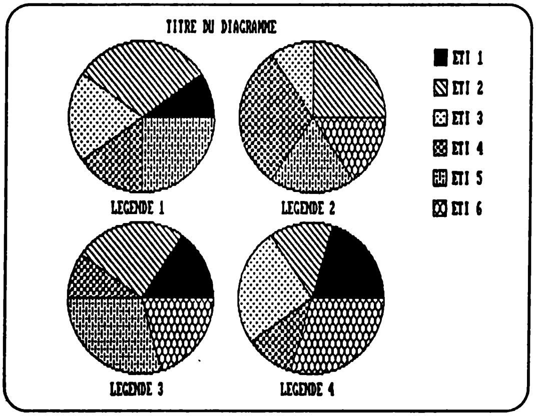


Fig. 5.12 : Diagramme comportant 4 camemberts

Et voilà le travail !... Vous voici désormais en possession du logiciel **DATAGRAF** de représentation graphique des données qui ne vous aura coûté que l'effort de l'avoir tapé de vos propres mains et qui, maléable à souhait, devrait pouvoir vous servir de tremplin vers l'élaboration d'autres diagrammes plus spécialisés.

Dans un diagramme à secteurs circulaires, chaque secteur est proportionnel au pourcentage que représente la donnée qui lui correspond par rapport à la somme des données.

Exercice sur le chapitre 5

1. Pourquoi les dimensions du repère orthogonal sont-elles de 560 × 160 points ?

   ☐ Pour que l'espacement entre valeurs soit un multiple entier de 48 pixels (6 caractères × 8 pixels/caractère)
   ☐ Pour que l'axe des Y puisse afficher 11 graduations équidistantes.
   ☐ Pour respecter le rapport hauteur/largeur de 5/12 en haute résolution.

2. Peut-on rajouter des valeurs supplémentaires de fin d'échelle au sous-programme de calcul automatique de l'échelle ?

   ☐ Oui, à condition de les placer par ordre croissant dans le sous-programme.
   ☐ Non, car il n'y a que 3 valeurs possibles ?

3. Comment empêche-t-on l'affichage d'une légende inutile ?

   ☐ En détectant la présence d'une donnée nulle dans la série.
   ☐ En court-circuitant les lignes d'affichage de la légende lorsque la somme des données d'une même série est nulle.

4. Quel est le rôle de la variable indicée SOMME (I) dans les diagrammes à aires superposées ?

   ☐ A dessiner une série en entier avant de passer à la suivante.
   ☐ A faire la somme des données d'une même étiquette.

5. Quelle est la largeur d'une colonne dans un diagramme à colonnes comportant 3 séries de 6 valeurs ?

   ☐ 48 pixels ☐ 60 pixels ☐ 72 pixels

6. Comment empêche-t-on le coloriage de déborder dans un diagramme à colonnes empilées ?

   ☐ En ne commandant le coloriage qu'à la condition que la hauteur de la colonne dépasse 1 pixel.
   ☐ En positionnant le point de début du coloriage sur le palier précédent.

7. Pourquoi avoir arrondi par défaut à 3.141592 la valeur de PI dans les diagrammes à secteurs circulaires ?

   ☐ Parce que c'est la valeur la plus proche de la valeur réelle.
   ☐ Pour que le dernier secteur puisse être dessiné.
   ☐ Pour éviter de « planter » le programme.

Réponses page 96

Solution des exercices sur le chapitre 4

1. A en simplifier l'écriture tout en le rendant plus lisible.
2. Non, car cette instruction en dispense.
3. Que chaque rubrique possède un numéro d'ordre.
4. Le contrôle de la longueur des entrées **ET** l'affichage du précédent contenu de la chaîne **ET** la gestion permanente des touches [ESC] et [TAB].
5. Bien que cela prenne moins de place sur la disquette, **c'est surtout** parce que cela permet de les modifier aussi souvent qu'on le désire.

Solution des exercices sur le chapitre 5

1. Pour que l'espacement entre valeurs soit un multiple entier de 48 pixels **ET** pour que l'axe des Y puisse afficher 11 graduations équidistantes.
2. Oui, à condition de les placer par ordre croissant dans le sous-programme.
3. En court-circuitant les lignes d'affichage de la légende lorsque la somme des données d'une même série est nulle.
4. A dessiner une série en entier avant de passer à la suivante **ET** à faire la somme des données d'une même étiquette.
5. 48 pixels.
6. En ne commandant le coloriage qu'à la condition que la hauteur de la colonne dépasse 1 pixel.
7. Pour que le dernier secteur soit dessiné **ET** pour éviter de planter le programme.

CHAPITRE VI

LES FORMES GRAPHIQUES

Les chapitres précédents vous ont décrit les méthodes et les instructions permettant d'obtenir des figures géométriques. Pour progresser dans l'art graphique, il vous faut maintenant aborder le dessin d'objets et de personnages dont la forme ne peut être ramenée à un simple assemblage de figures géométriques élémentaires. C'est ce que l'on nomme les formes graphiques. Or, totalement inadaptées à la création de telles formes, les instructions graphiques que vous connaissez vont devoir céder la place à un véritable sous-langage graphique spécialement conçu pour elles. Le rôle du présent chapitre est de vous décrire ce nouveau sous-langage, ainsi que les méthodes d'emploi qui s'y rattachent.

Menu du chapitre 6

Le sous-langage DRAW
Règles de codage d'une forme graphique
Sauvegarde et chargement d'une forme graphique
Constitution d'une bibliothèque de formes
FUSFORM : Un gestionnaire de fichier de formes.

Quel est votre choix : ■

LE SOUS-LANGAGE DRAW

Bien que les instructions graphiques **COLOR, LINE, CIRCLE** et **PAINT** puissent théoriquement dessiner un objet ou un personnage constitué d'une grande quantité de pixels de diverses couleurs, il n'en demeure pas moins que le résultat risque d'être franchement décevant sur le plan pratique, tant sur le plan du volume du sous-programme qu'une telle démarche nécessite, que sur celui de son temps d'exécution qui est franchement loin d'être rapide.

DRAW

C'est pourquoi le Basic Microsoft vous propose un sous-langage graphique spécialement adapté à cette fonction et auquel on accède à travers l'instruction **DRAW** (*). Ce sous-langage comporte toute une série de codes à une lettre qui, insérés au sein d'une chaîne, vont vous permettre de dessiner des points ou des traits horizontalement, verticalement ou en diagonale, de fixer la couleur de chaque pixel si le besoin s'en fait sentir, de faire tourner la forme, de la rendre plus grande ou plus petite comme avec un zoom, etc, etc... Bref, un super langage graphique aux possibilités innombrables !

(*) La fiche détaillée de l'instruction figure à la page 312 du volume 2 de Pratique des IBM-PC et compatibles.

Déplacements

Les codes de déplacement, au nombre de huit, sont résumés dans le tableau 6.1. que voici :

Lettre-code	Déplacement
Un	Vers le haut
Dn	Vers le bas
Ln	Vers la gauche
Rn	Vers la droite
En	En diagonale vers le haut à droite
Fn	En diagonale vers le bas à droite
Gn	En diagonale vers le bas à gauche
Hn	En diagonale vers le haut à gauche
L'indice n précise le nombre de pixels du déplacement	

Tableau 6.1 : Codes de déplacement du sous-langage DRAW

Pour se servir des codes de déplacement, il suffit de les placer dans une chaîne en faisant suivre chacun d'eux de l'indication du nombre de pixels du déplacement (c'est l'indice n du tableau). C'est ainsi que pour obtenir un carré de 9 pixels de côté, vous pouvez écrire :

Fixation du point de référence...

```
10 KEY OFF
20 SCREEN 1:CLS:COLOR 8,0
30 X=160:Y=100:PSET(X,Y),1
40 DRAW "R8 D8 L8 U8"
```

... et dessin du carré !

... ou encore, en attribuant un nom à la chaîne de codes :

En attribuant un nom à la chaîne...

```
10 KEY OFF
20 SCREEN 1:CLS:COLOR 8,0
30 CARRE$="R8 D8 L8 U8"
40 X=160:Y=100:PSET(X,Y),1
50 DRAW CARRE$
```

... on sait ce que DRAW dessine !

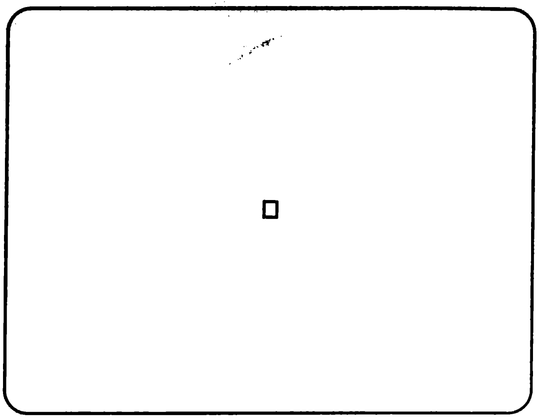


Fig. 6.1 : Affichage d'une forme carrée.

Ce premier essai appelle quelques commentaires. Tout d'abord, les espaces qui séparent les codes l'un de l'autre sont totalement inutiles. Ils ne sont (provisoirement) là que pour des raisons de facilité de lecture (pour le lecteur, s'entend !). Ensuite, il peut paraître surprenant d'obtenir un carré de 9 pixels de côté avec seulement des déplacements de 8 pixels. Cela provient tout simplement du fait que ceux-ci s'opèrent en mode relatif, ce qui fait que partant du pixel numéro n, on aboutit au pixel n + 8 après un déplacement de 8 pixels, soit un total de 8 + 1 = 9 pixels.

Remarque : *Il est inutile de préciser le chiffre 1 à la suite d'une lettre-code si l'on cherche à obtenir un déplacement d'un pixel. Dans le même ordre d'idées, il est possible de dessiner un point isolé en faisant suivre sa lettre-code du chiffre 0.*

Déplacements sans affichage

Lorsqu'on dessine une forme graphique, il faut pouvoir de temps en temps « lever le crayon » pour aller se positionner ailleurs, sans pour autant laisser une trace derrière soi. C'est ce que permet le sous-langage DRAW à travers les 2 codes suivants :

Lettre-code	Action
B	Déplacement sans affichage.
N	Déplacement avec affichage suivi d'un retour à la position précédente.

Tableau 6.2 : Codes de déplacement sans affichage et de retour à la position précédente.

Afin d'illuster les déplacements sans affichage, voici un exemple de dessin d'une cible de pointage :

Déplacements sans affichage...

```
10 KEY OFF
20 SCREEN 1:CLS:COLOR 8,1
30 CIBLE$="BU2 NU2 BF2 NR2 BG2 ND2 BH2 NL2"
40 X=160:Y=100:PSET(X,Y),2
50 DRAW CIBLE$
```

... ou avec retour à la position précédente !

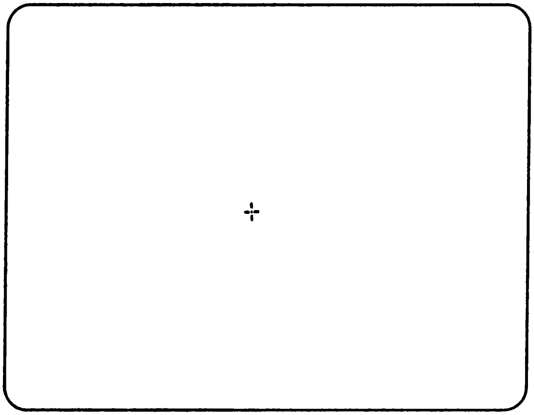


Fig. 6.2 : Dessin d'une cible comportant des déplacements sans affichage

Il est bon de noter que l'instruction **PSET** fixe à la fois le point de référence et la couleur de la cible. Or, ceci présente un inconvénient majeur dans la mesure où le point de référence se trouve de ce fait systématiquement affiché dans la couleur de la cible, même si l'on souhaite qu'il ne le soit pas !

Fixation de la couleur

Une solution élégante à ce problème consiste à fixer la couleur, non plus à l'aide de l'instruction **PSET**, mais en se servant d'une lettre-code du sous-langage **DRAW**. Ainsi, le point de référence peut très bien être fixé avec un **PRESET (X, Y)**, ce qui évite qu'il soit dessiné systématiquement.

Lettre-code	Action
Cn	Fixe la couleur
n est compris entre 0 et 3 en moyenne résolution. n est égal à 0 ou 1 en haute résolution.	

Si l'on reprend l'exemple précédent, il est possible de le ré-écrire comme suit en employant la lettre-code Cn :

Fixation de la couleur de la forme...

```
10 KEY OFF
20 SCREEN 1:CLS:COLOR,1
30 CIBLE$="C2 BU2 NU2 BF2 NR2 BG2 ND2 BH2 NL2"
40 X=160:Y=100:PRESET(X,Y)
50 DRAW CIBLE$
```

... et positionnement d
son point de référence :

Comme vous pouvez le constater « de visu », cette nouvelle forme d'écriture évite l'apparition « fâcheuse » du point de référence de la forme :

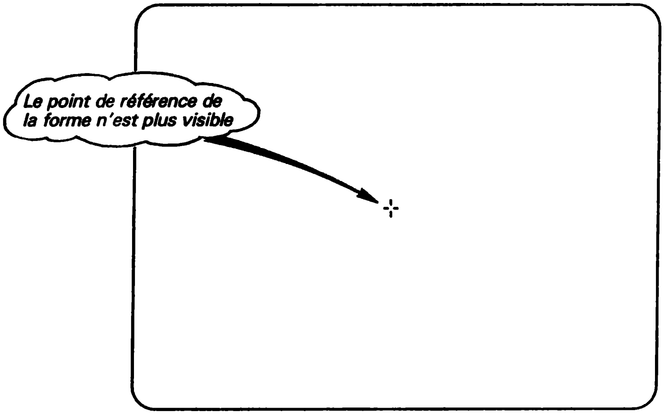


Fig. 6.3 : L'emploi de la lettre-code Cn évite l'affichage systématique du point de référence de la forme.

Agrandissement et réduction

Le sous-langage **DRAW** dispose d'un facteur d'échelle qui est déterminé par la lettre-code S suivie d'un nombre compris entre 1 et 255.

Lettre-code	Action
Sn	Définit l'échelle de reproduction
n peut être compris entre 1 et 255. Par défaut, l'échelle est fixée à 4.	

Grâce à lui, il va donc être possible d'agrandir ou de réduire la taille des formes graphiques. La valeur par défaut étant fixée à 4, il s'en suit que le cœfficient réducteur maximal est de 1/4 et le cœfficient multiplicateur maximal voisin de 64. C'est tout de même pas mal, n'est-ce pas ?

Pour vérifier si les choses se passent bien ainsi, essayez donc le petit programme que voci qui dessine un carré de plus en plus grand :

Fixation de l'échelle

```
10 KEY OFF
20 SCREEN 1:CLS:COLOR 8,0
30 FOR I=1 TO 64 STEP 3
40 CARRE$="S=I;C1 BH4 R8 D8 L8 U8"
50 X=160:Y=100:PRESET(X,Y)
60 DRAW CARRE$
70 NEXT
```

Pratique, n'est-ce pas ?... Au fait, avez-vous remarqué que le sous-langage **DRAW** accepte d'affecter des variables aux lettres-code. En l'occurence, c'est la variable I qui fixe l'échelle de la forme en lieu et place de l'indice n. Notez au passage la présence **obligatoire** d'un point virgule en tant que délimitateur d'une opération d'affectation au sein d'une chaîne de codes (lignes 40).

Remarque : *L'échelle 1 correspondant à l'indice 4, il s'en suit qu'il faut diviser par 4 les valeurs d'indice pour obtenir la valeur de l'échelle. Ainsi, un indice 8 correspond à l'échelle 2 (8/4).*

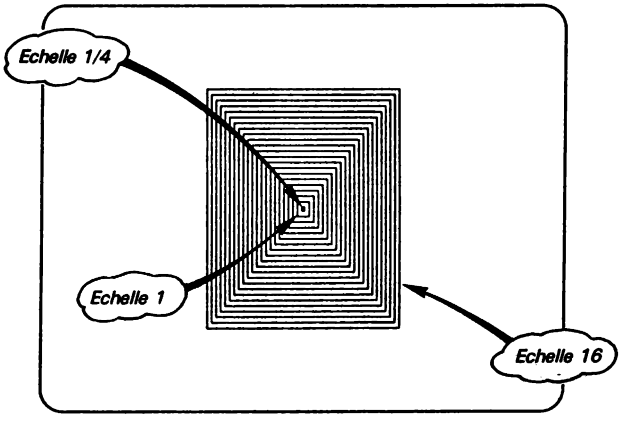


Fig. 6.4 : Carrés dessinés de l'échelle 1/4 à l'échelle 16.

Rotation autour d'un point

Il est également possible de faire pivoter une forme graphique de 1 à 360 degrés autour de son point de référence à l'aide des lettres-code du tableau suivant :

Lettre-code	Action
An	Etablit un angle de rotation égal à n. 90 degrés.
TA degrés	Etablit tout angle de rotation compris entre - 360 et 360 degrés.
n est compris entre 0 et 3	

Comme vous pouvez le constater, il existe deux lettres-code différentes selon que vous souhaitez faire pivoter la forme d'un multiple de 90 degrés (An) ou de toute valeur comprise entre - 360 et 360 degrés (TA degrés). Quant au sens de rotation, il s'agit du sens trigonométrique habituel, tel que l'illustre le petit bout de programme que voici :

```
10 KEY OFF
20 SCREEN 1:CLS:COLOR 8,0
30 FOR I=0 TO 3
40 FLECHE$="A=I;C3 BR32 G4 U2 L12 U4 R12 U2 F4"
50 X=160:Y=100:PRESET(X,Y)
60 DRAW FLECHE$
70 FOR T=1 TO 500:NEXT
80 NEXT
```

Cette boucle... (ligne 30)

... fait pivoter cette flèche de 90 degrés en 90 degrés dans le sens trigonométrique (ligne 40)

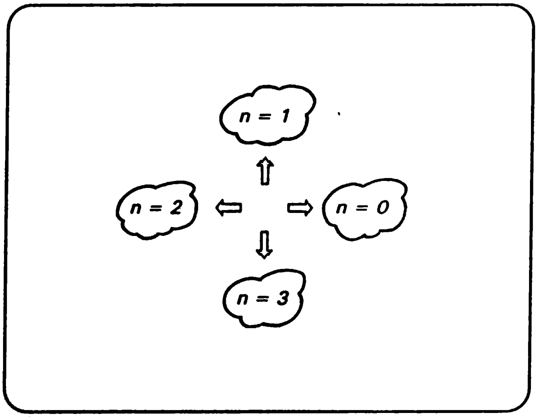


Fig. 6.5 : Rotation d'une forme graphique

> **Le sous-langage DRAW permet de dessiner ou d'effacer une forme graphique, d'en fixer la couleur, de l'agrandir, de la réduire et de la faire pivoter sur elle-même.**

RÈGLES DE CODAGE D'UNE FORME GRAPHIQUE

Vous voici désormais à la tête d'un véritable langage graphique parfaitement adapté aux formes, objets et personnages qui vont bientôt faire leur apparition dans vos logiciels. Pourtant, savez-vous coder de telles formes graphiques ?... Rien n'est moins sûr, car ce ne sont pas les quelques exemples précédents (trop simple au demeurant) qui peuvent donner l'exacte mesure de votre habileté à coder une forme complexe.
Nous allons donc examiner en détails le processus de codage d'une forme assez élaborée, telle que celle de l'exemple que voici :

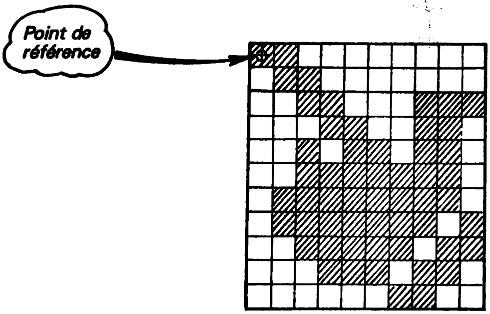


Fig. 6.6 : Main dont l'index sert à pointer sur une case-menu.

Il s'agit d'une main que l'on déplace sur l'écran (généralement à l'aide de la souris) et qui permet de sélectionner une option dans un menu. Intéressante, n'est-ce pas !

Maintenant que l'on dispose du dessin de la forme, la question se pose de savoir par quel bout le prendre. Où doit-on débuter le codage ?... Mais par son point de référence, c'est-à-dire par celui dont on fixe les coordonnées à l'aide d'un **PRESET** ou d'un **PSET** préalablement à l'appel de l'instruction **DRAW**. Et où doit-on placer un point de référence ?... En fait, cela dépend, non pas tant de la forme, mais de l'usage que l'on en fait !

Si la forme est destinée à pivoter autour d'un point fixe (pale d'hélicoptère, aile de moulin, etc, etc...), il est plus judicieux, car cela simplifie la programmation du mouvement, de confondre le point de référence avec le point de rotation. Si, par contre, la forme n'est destinée qu'à subir de simples translations horizontales ou verticales, l'usage veut (mais rien n'empêche de le changer) que l'on choisisse le point supérieur gauche, comme indiqué sur la figure 6.6.

Ce premier point étant acquis, il faut maintenant procéder au codage des différents pixels allumés que nécessite cette forme. Comment doit-on procéder ?... Faut-il analyser la forme ligne par ligne, colonne par colonne ou selon l'inspiration du moment ?... Face à un tel choix, une seule réponse s'impose : choisir la méthode qui fait appel au minimum de lettres-code. De cette façon, l'on est sûr d'afficher la forme dans un minimum de temps.

Le codage ligne par ligne ou colonne par colonne étant équivalent et l'usage en ce domaine étant de coder une forme ligne par ligne, nous allons nous orienter vers cette méthode de codage.

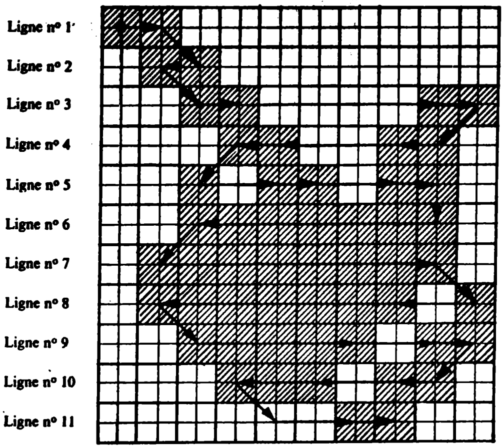


Fig. 6.7 : Codage ligne par ligne des pixels allumés d'une forme.

Vous noterez que les lignes sont tantôt parcourues dans un sens ou dans l'autre, les lignes impaires l'étant de gauche à droite et les lignes paires de droite à gauche. De cette façon, vous limitez les déplacements à leur strict minimum.

Voici d'ailleurs le résultat du codage de cette main :

Lettres-code de la main

```
10 KEY OFF
20 SCREEN 1:CLS:COLOR 8,0
30 MAIN$="C3RFLFRBR5RGLBL3LGBR2RBR2RDL6GR7FBL2L6FR4BR2RGLBL2L2BFBR2R"
40 X=160:Y=100:PSET(X,Y)
50 DRAW MAIN$
```

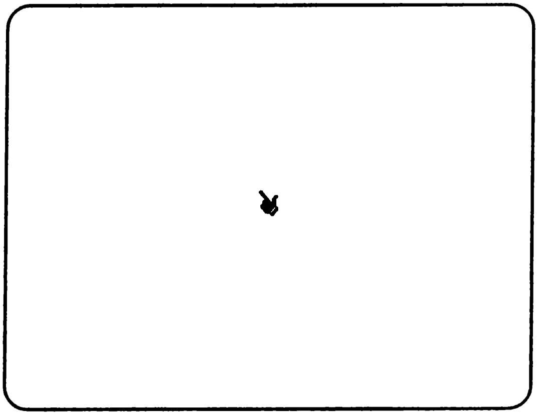


Fig. 6.8 : Affichage de la main après codage ligne par ligne.

Théoriquement, le codage ligne par ligne avec alternance du sens de parcours (lignes paires ou impaires) est celui qui nécessite le minimum de déplacements. Essayez donc de faire mieux avec une autre méthode !

> **Le codage d'une forme graphique débute par son point de référence et s'effectue ligne par ligne en alternant le sens du parcours selon que la ligne est paire ou impaire.**

SAUVEGARDE ET CHARGEMENT D'UNE FORME GRAPHIQUE

Vous venez de terminer le codage d'une forme graphique assez compliquée et vous aimeriez bien la sauvegarder sur disquette de façon à pouvoir l'utiliser de nouveau dans d'autres programmes. Comment faut-il procéder ?
La première idée qui vient à l'esprit consiste à traiter la forme graphique comme s'il s'agissait d'une chaîne (ce qu'elle est en fait) et à la sauvegarder dans un fichier séquentiel. A titre d'exemple, voici comment il vous faut procéder pour sauvegarder la main que vous venez de coder précédemment :

```
10 MAIN$="C3RFLFRBR5RGLBL3LGBR2RBR2RDL6GR7FBL2L6FR4BR2RGLBL2L2BFBR2R"
20 FICHIER$="MAIN"
30 OPEN FICHIER$ FOR OUTPUT AS#1
40 WRITE#1,MAIN$:CLOSE#1
```

Sauvegarde d'une forme graphique (lignes 30 et 40)

Comme vous pouvez le constater à l'usage, la sauvegarde est toute simple à mettre en œuvre. Il en est de même, d'ailleurs, du chargement d'une forme dans un programme :

```
10 FICHIER$="MAIN"
20 OPEN FICHIER$ FOR INPUT AS#1
30 INPUT#1,MAIN$:CLOSE#1
40 KEY OFF
50 SCREEN 1:CLS:COLOR 8,0
60 X=160:Y=100:PRESET(X,Y)
70 DRAW MAIN$
```

Chargement d'une forme graphique... (lignes 10 à 30)

... suivi de son affichage au centre de l'écran ! (lignes 40 à 70)

Ce dernier en effet ne nécessite que les lignes 10 à 30, les lignes 40 à 70 servant à afficher la forme graphique en plein centre de l'écran.

Et si au lieu d'une seule forme graphique, le programme en nécessite plusieurs ?... Là, il faut bien avouer que la situation se complique un peu dans la mesure où chaque forme graphique est incluse dans un fichier qui lui est propre. Une première solution (pas géniale pour deux sous...) consiste à répéter les lignes 10 à 30 autant de fois qu'il y a de formes à charger. En supposant qu'il y en ait 4, on aboutit au programme (volumineux) que voici :

```
10 FICHIER$="CARRE"
20 OPEN FICHIER$ FOR INPUT AS#1
30 INPUT#1,CARRE$:CLOSE#1
40 FICHIER$="CIBLE"
50 OPEN FICHIER$ FOR INPUT AS#1
60 INPUT#1,CIBLE$:CLOSE#1
70 FICHIER$="FLECHE"
80 OPEN FICHIER$ FOR INPUT AS#1
90 INPUT#1,FLECHE$:CLOSE#1
100 FICHIER$="MAIN"
110 OPEN FICHIER$ FOR INPUT AS#1
120 INPUT#1,MAIN$:CLOSE#1
130 KEY OFF
140 SCREEN 1:CLS:COLOR 8,0
150 X=80:Y=50:PRESET(X,Y):DRAW CARRE$
160 X=240:Y=50:PRESET(X,Y):DRAW CIBLE$
170 X=80:Y=150:PRESET(X,Y):DRAW FLECHE$
180 X=240:Y=150:PRESET(X,Y):DRAW MAIN$
```

Inutile de dire à quel point un tel programme est lourd !... Une deuxième solution (un peu plus élégante, mais guère mieux...) consiste à placer les lignes de chargement de chaque fichier de forme dans un sous-programme. Bien entendu, cela implique que l'on abandonne les appellations personnalisées des formes (comme carré, cible, flèche, etc.) pour leur préférer d'autres plus sobres telles que **FORMES$(I)**. On aboutit alors à l'écriture suivante :

Le chargement des fichiers de forme s'opère...

```
10 DIM FORME$(4)
20 FICHIER$="CARRE":I=1:GOSUB 140
30 FICHIER$="CIBLE":I=2:GOSUB 140
40 FICHIER$="FLECHE":I=3:GOSUB 140
50 FICHIER$="MAIN":I=4:GOSUB 140
60 KEY OFF
70 SCREEN 1:CLS:COLOR 8,0
80 X=80:Y=50:PRESET(X,Y):DRAW FORME$(1)
90 X=240:Y=50:PRESET(X,Y):DRAW FORME$(2)
100 X=80:Y=150:PRESET(X,Y):DRAW FORME$(3)
110 X=240:Y=150:PRESET(X,Y):DRAW FORME$(4)
120 END
130 'Sous-programme de chargement------------------------------------------
140 OPEN FICHIER$ FOR INPUT AS#1
150 INPUT#1,FORME$(I):CLOSE#1
160 RETURN
```

... grâce au sous-programme que voici !

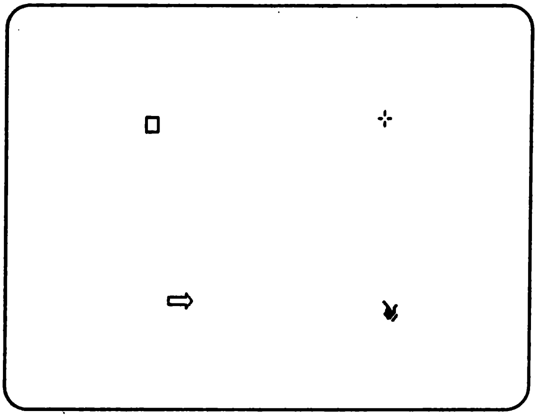


Fig. 6.9 : Chargement et affichage de plusieurs formes graphiques.

Bien que légèrement supérieure à la précédente, cette solution n'est toutefois valable qu'à la condition que le nombre de fichiers à charger soit faible (tout au plus 2 ou 3). Au-delà, cela n'a plus de sens !... Imaginez qu'un programme nécessite une centaine de formes (si, si, c'est possible). Allez-vous les charger les unes à la suite des autres ?... Non, bien sûr !... Vous allez pour cela faire appel à une bibliothèque de formes.

> **Chaque forme graphique pouvant être considérée comme une chaîne de caractères, il est possible de la sauvegarder ou de la charger à partir d'un fichier séquentiel qui lui est personnel.**

CONSTITUTION D'UNE BIBLIOTHÈQUE DE FORMES

Qu'est-ce donc qu'une bibliothèque de formes ?... C'est un ensemble de formes graphiques regroupées en un même fichier séquentiel. Ainsi, que le fichier en comporte dix, cent ou plus encore, il n'y a en tout et pour tout qu'un seul fichier à charger. Vous imaginez facilement le gain de temps et de place que cela procure.

Pour vous en convaincre, essayez donc les deux petits programmes suivants. Le premier a pour objet de sauvegarder les 4 formes précédentes (pour le moment, il n'y en a pas d'autres) en un fichier séquentiel unique :

```
10 CARRE$="C1L8D8R8U8"
20 CIBLE$="C2BU2NU2BF2NR2BG2ND2BH2NL2"
30 FLECHE$="C3G4U2L12U4R12U2F4"
40 MAIN$="C3RFLFRBR5RGLBL3LGBR2RBR2RDL6GR7FBL2L6FR4BR2RGLBL2L2BFBR2R"
50 FICHIER$="FORMES"
60 OPEN FICHIER$ FOR OUTPUT AS#1
70 WRITE#1,CARRE$,CIBLE$,FLECHE$,MAIN$:CLOSE#1
```

Ce fichier unique...

... contient ces 4 formes !

... et le second de les recharger en vue de leur affichage sur l'écran :

Chargement du fichier de formes...

```
10 FICHIER$="FORMES"
20 OPEN FICHIER$ FOR INPUT AS#1
30 INPUT#1,CARRE$,CIBLE$,FLECHE$,MAIN$:CLOSE#1
40 KEY OFF
50 SCREEN 1:CLS:COLOR 8,0
60 X=80:Y=50:PRESET(X,Y):DRAW CARRE$
70 X=240:Y=50:PRESET(X,Y):DRAW CIBLE$
80 X=80:Y=150:PRESET(X,Y):DRAW FLECHE$
90 X=240:Y=150:PRESET(X,Y):DRAW MAIN$
```

... et affichage des formes qu'il contient !

Avez-vous remarqué à quelle vitesse s'effectue le chargement ?... En un temps record !... De surcroît, les formes conservent leur nom, ce qui, il faut bien l'avouer, est plus pratique que de devoir se souvenir du contenu des chaînes indicées **FORMES$(I)** comme c'était le cas précédemment.

Ainsi donc, vous allez vous orienter systématiquement vers la création d'un fichier de formes, quel que soit le nombre de formes que votre programme emploie. C'est pour vous l'assurance d'une écriture aisée associée à un chargement ultra-rapide.

L'ennui, car il y en a un beau, c'est qu'un tel fichier de formes est destiné à subir diverses modifications telles qu'adjonction ou suppression de formes, changement de places au sein du fichier, etc. Or, comment réaliser de telles modifications si ce n'est à l'aide d'un gestionnaire de fichier de formes ?... C'est pourquoi vous allez découvrir au fil des pages suivantes la description détaillée du logiciel **FUSFORM** spécialement adapté à cette fonction de gestion.

Rassemblées en une bibliothèque au sein d'un fichier de formes unique, les formes graphiques sont rapidement chargées et disponibles au sein d'un programme Basic.

FUSFORM : UN GESTIONNAIRE DE FICHIERS DE FORMES

La création d'un gestionnaire de fichiers de formes nécessite que soient définies de manière très précise les caractéristiques des fichiers de formes qu'il devra gérer. C'est pourquoi il est bon de se pencher quelques instants sur ce point important avant de débuter l'étude du gestionnaire **FUSFORM**.

La première question que l'on doit se poser lorsque l'on est en présence d'un fichier de formes (en fait, c'est le gestionnaire de fichiers qui se la pose...) est de savoir de combien de formes ce fichier est constitué. Or, la première idée qui vient à l'esprit pour signaler le nombre de formes que contient un fichier est d'incorporer ce nombre dans le nom du fichier. On dispose pour cela des 3 caractères qui, placés après un point de séparation, servent à préciser la nature de chaque fichier (BAS, COM, EXE, BAT, etc...). Ainsi, d'un seul coup d'œil au répertoire, vous serez à même de déterminer que le fichier **FORMES.4** est composé de 4 formes. La méthode est simple, directe et efficace !

Ce premier point étant réglé, passons au suivant !... Comment doit-on procéder pour déterminer le nom des différentes formes contenues dans une bibliothèque de formes ?... Si à priori l'on peut penser qu'un programme employant un fichier de formes en connaît les noms et l'ordre dans lequel elles sont disposées, il n'en est pas de même de la part d'un gestionnaire de fichiers. Il y a donc lieu d'incorporer le nom de chaque forme au sein du fichier de formes.

Pour cela, il existe plusieurs solutions. Celle qui a été retenue (mais rien ne vous empêche d'en changer) consiste à placer le nom d'une forme devant celle-ci. Le fichier de formes est alors constitué d'une succession alternée de noms et de formes telle que l'illustre la figure 6.10 :

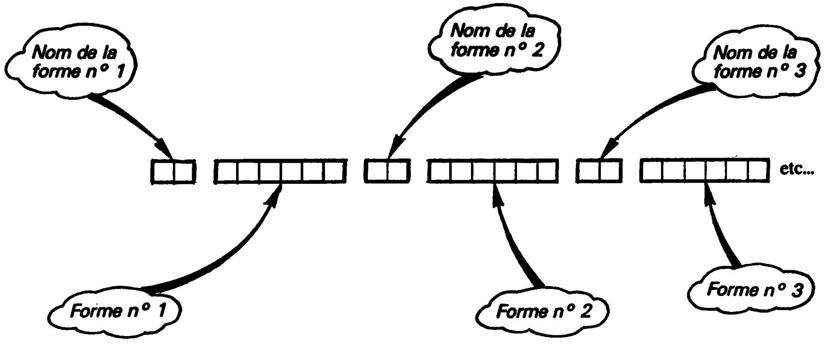


Fig. 6.10 : Constitution d'un fichier de formes

FUSFORM

Ce point important étant acquis, il est temps maintenant d'aborder l'étude du gestionnaire **FUSFORM**. Quelles sont les fonctions qu'il doit accomplir et les commandes qui s'y rattachent ?... Dans l'ordre de leur présentation à l'écran, ce sont :

LIST	Permet de lister tout ou partie du contenu d'un fichier de formes en indiquant pour chaque forme son nom et sa position au sein du fichier.
RENAME	Attribue un nouveau nom à une forme.
LOAD"	Assure le chargement d'un fichier de formes.
SAVE"	Assure la sauvegarde d'un fichier de formes.
INSERT	Réalise l'insertion d'une forme nouvelle au sein d'un fichier de formes.
DELETE	Supprime une forme au sein d'un fichier de formes.
SWAP	Permute la position de deux formes au sein d'un fichier de formes.
FILES	Affiche le répertoire de la disquette.
QUIT	Permet de quitter le gestionnaire **FUSFORM**.

Comme vous pouvez le voir, il y a du pain sur la planche !... Si donc l'effort ne vous fait pas peur (et si le gestionnaire FUSFORM présente un intérêt pour vous...), faites craquer vos articulations des doigts et au boulot !

Comme tout programme, celui-ci débute par le dimensionnement des variables (ligne 10), la préparation de l'écran de travail (lignes 20 et 30) et l'affectation des touches de fonction (lignes 40 et 50) :

```
10 DIM NOM$(255),FORME$(255)
20 SCREEN 0:COLOR 15,1,1:CLS
30 LOCATE 1,32:PRINT"*** FUSFORM ***"
40 KEY 2,"RENAME ":KEY 5,"INSERT ":KEY 6,"DELETE ":KEY 7,"SWAP "
50 KEY 8,"FILES"+CHR$(13):KEY 9,"QUIT"+CHR$(13):KEY 10,""
```

Ceci fait, on passe au menu principal dont le rôle est d'assurer l'aiguillage vers les sous-programmes de traitement des fonctions. Notez au passage que la méthode employée permet la frappe des commandes au clavier comme à partir des touches de fonction :

```
60 'Menu principal-----------------------------------------------------------------
70 PRINT:INPUT ENTREE$
80 IF LEFT$(ENTREE$,4)="LIST" THEN 240
90 IF LEFT$(ENTREE$,6)="RENAME" THEN 420
100 IF LEFT$(ENTREE$,4)="LOAD" THEN 470
110 IF LEFT$(ENTREE$,4)="SAVE" THEN 700
120 IF LEFT$(ENTREE$,6)="INSERT" THEN 880
130 IF LEFT$(ENTREE$,6)="DELETE" THEN 1000
140 IF LEFT$(ENTREE$,4)="SWAP" THEN 1080
150 IF LEFT$(ENTREE$,5)="FILES" THEN FILES:GOTO 70
160 IF LEFT$(ENTREE$,4)="QUIT" THEN CLS:END
170 PRINT TAB(3)"Erreur de syntaxe !":BEEP:GOTO 70
180 PRINT TAB(3)"Quantité illégale !":BEEP:GOTO 70
190 PRINT TAB(3)"Nom de fichier incorrect !":BEEP:GOTO 70
200 PRINT TAB(3)"Ce fichier n'existe pas !":BEEP:RESUME 220
210 PRINT TAB(3)"Disquette protégée contre l'écriture !":BEEP:RESUME 220
220 ON ERROR GOTO 0:GOTO 70
```

Si votre commande ne figure pas dans cette liste...

... vous obtenez ce message d'erreur !

Voyons un peu comment tout cela fonctionne !... La chaîne **ENTRÉE$** après saisie à la ligne 70 est analysée de façon à y rechercher l'une des 9 commandes disponibles (lignes 80 à 160). Si la recherche aboutit, le branchement s'effectue (via un **GOTO** et non un **GOSUB**) vers le sous-programme approprié. Dans le cas contraire, la ligne 170 vous gratifie d'un message du type...

ERREUR DE SYNTAXE

... qui, en français cette fois-ci, ne laisse planer aucun doute sur votre incapacité à respecter les élémentaires règles de syntaxe du gestionnaire **FUSFORM** (qui, soit dit en passant, sont les mêmes que celles du Basic). Vous noterez d'ailleurs que ce message d'erreur n'est pas le seul et que toute une collection vous attend (lignes 180 à 220) au fil des diverses commandes que comporte ce gestionnaire de fichiers.

LIST

Parlant de commandes, voici la première d'entre-elles. Il s'agit de la commande **LIST** qui permet de lister tout ou partie du contenu d'un fichier de formes :

```
230 'Routine LIST------------------------------------------------------------
240 IF NOMBRE=0 THEN 70
250 LONGUEUR=LEN(ENTREE$)
260 IF LONGUEUR<5 THEN DEBUT=1:FIN=NOMBRE:GOTO 370
270 DELIMITEUR=0:FOR I=5 TO LONGUEUR
280 IF MID$(ENTREE$,I,1)="-" THEN DELIMITEUR=DELIMITEUR+1:
    IF DELIMITEUR=1 THEN POSITION=I
290 NEXT:IF DELIMITEUR>1 THEN 170
300 IF DELIMITEUR=0 THEN DEBUT=VAL(RIGHT$(ENTREE$,LONGUEUR-4)):
    FIN=DEBUT:IF DEBUT=0 OR DEBUT>NOMBRE THEN 180 ELSE 370
310 IF POSITION-5<1 THEN DEBUT=1:GOTO 350
320 DEBUT=VAL(MID$(ENTREE$,5,POSITION-5))
330 IF DEBUT=0 THEN DEBUT=1:GOTO 350
340 IF DEBUT<1 OR DEBUT>NOMBRE THEN 180
350 IF LONGUEUR-POSITION<1 THEN FIN=NOMBRE:GOTO 370
360 FIN=VAL(RIGHT$(ENTREE$,LONGUEUR-POSITION))
370 IF DEBUT>FIN THEN 170
380 IF FIN>NOMBRE THEN 180
390 FOR I=DEBUT TO FIN:PRINT TAB(11) I;TAB(16) NOM$(I):NEXT
400 GOTO 70
```

Cette routine peut vous paraître bien longue pour, somme toute, ne lister qu'une suite de chaînes. En fait, le listage se résume à une seule ligne (la 390), le reste du sous-programme étant destiné à déterminer les numéros des formes à partir desquelles débute et se termine le listage. Pour information, sachez que la commande **LIST** peut s'employer seule, ou suivie par un numéro de ligne, ou suivie par un numéro de ligne lui-même précédé ou suivi d'un trait d'union, ou encore suivie par deux numéros de lignes séparés par un trait d'union.

C'est ainsi qu'après avoir déterminé la **LONGUEUR** de la chaîne (ligne 250), le sous-programme recherche la présence d'un **DELIMITEUR** au sein de celle-ci (lignes 270 à 290). Lorsqu'il en découvre un, il note sa **POSITION** (lignes 280) et s'en sert pour effectuer la lecture du numéro de ligne du **DEBUT** du listage (lignes 310 et 320) et de **FIN** (lignes 360).

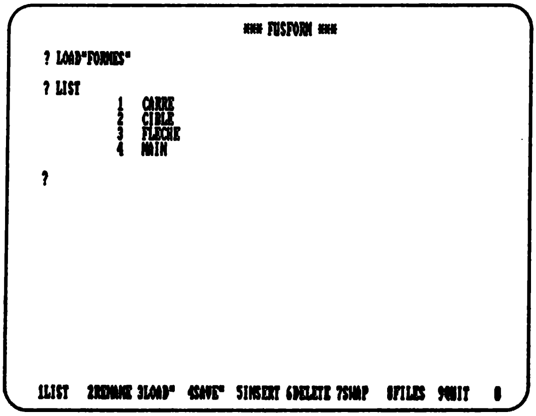


Fig. 6.11 : Listage du contenu d'un fichier de formes.

Bien entendu, toute cette recherche est agrémentée de tests visant à déterminer si le **NOMBRE** de formes à lister est supérieur à zéro (ligne 240), si le **DELIMITEUR** est présent ou non (ligne 300), si la valeur de **DEBUT** est présente (ligne 330) et si elle n'excède pas le **NOMBRE** de formes à lister (lignes 340), si la valeur de **FIN** est également présente (ligne 350) et si, elle non plus, n'excède pas le **NOMBRE** de formes à lister (ligne 380), tout en s'assurant qu'elle est plus élevée que la valeur de **DEBUT** (ligne 370)

RENAME

La routine suivante à passer sur la sellette et celle de la commande **RENAME** :

```
410 'Routine RENAME------------------------------------------------------
420 LONGUEUR=LEN(ENTREE$):IF LONGUEUR<7 THEN 170
430 I=VAL(RIGHT$(ENTREE$,LONGUEUR-6)):IF I=0 THEN 170
440 IF I>NOMBRE THEN 180
450 PRINT TAB(3)"Nouveau nom: ";:INPUT"",NOM$(I)
460 GOTO 70
```

Comme vous pouvez le remarquer, elle se réduit à très peu de choses. La ligne 420 détermine la **LONGUEUR** de la chaîne d'**ENTRÉE** et renvoie vers le message d'erreur de syntaxe (ligne 170) chaque fois qu'elle constate l'absence d'un numéro de forme à la suite de la commande **RENAME**. Ceci fait, la ligne 430 extrait de la chaîne le **NUMERO** de la forme dont il convient de changer le nom et, si celui-ci reste dans les limites du **NOMBRE** de formes disponibles (ligne 440), la ligne suivante prend le relais (ligne 450) pour vous demander le nouveau nom qu'il convient d'attribuer à cette forme.

LOAD

Au tour maintenant de la routine **LOAD** qui assure le chargement d'un fichier de formes :

Détermination du nom du fichier...

```
470 'Routine LOAD--------------------------------------------------------
480 LONGUEUR=LEN(ENTREE$):IF LONGUEUR<5 THEN 170
490 DELIMITEUR=0:FOR I=5 TO LONGUEUR
500 IF MID$(ENTREE$,I,1)=CHR$(34) THEN GOSUB 650
510 NEXT:IF DELIMITEUR=0 OR DELIMITEUR>2 THEN 170
520 IF PREMIER=LONGUEUR OR DEUXIEME-PREMIER<2 THEN 190
530 FICHIER$=MID$(ENTREE$,PREMIER+1,DEUXIEME-PREMIER-1)
540 LONGUEUR=LEN(FICHIER$)
550 DELIMITEUR=0:FOR I=1 TO LONGUEUR
560 IF MID$(FICHIER$,I,1)="." THEN DELIMITEUR=DELIMITEUR+1:
    IF DELIMITEUR=1 THEN POSITION=I
570 NEXT:IF DELIMITEUR>1 OR POSITION=1 OR LONGUEUR-POSITION<1 THEN 190
580 IF DELIMITEUR=0 THEN NOMBRE=1:GOTO 610
590 NOMBRE=VAL(RIGHT$(FICHIER$,LONGUEUR-POSITION))
600 IF NOMBRE=0 OR NOMBRE>255 THEN 190
610 ON ERROR GOTO 200
620 OPEN FICHIER$ FOR INPUT AS#1
630 FOR I=1 TO NOMBRE:INPUT#1,NOM$(I),FORME$(I):NEXT
640 CLOSE#1:GOTO 220
650 DELIMITEUR=DELIMITEUR+1
660 IF DELIMITEUR=1 THEN PREMIER=I:DEUXIEME=LONGUEUR+1
670 IF DELIMITEUR=2 THEN DEUXIEME=I
680 RETURN
```

... et chargement de celui-ci !

Bien que longue, cette routine ne fait que réemployer des structures identiques (ou presque...) à celles des routines précédentes. C'est ainsi qu'après avoir déterminé la présence de guillemets (code CHR$(34) de la ligne 500), le nom du **FICHIER** est ensuite analysé de façon à repérer la **POSITION** du point de séparation (lignes 550 à 580). Ce faisant, il suffit de lire le **NOMBRE** situé à la suite de ce **DELIMITEUR** pour connaître le **NOMBRE** de formes contenues dans le fichier (lignes 590 et 600).

En ce qui concerne le chargement du fichier à proprement parler, vous noterez la ligne de renvoi sur erreur (ligne 610) qui entre en jeu chaque fois que vous stipulez un nom du fichier qui n'existe pas sur la disquette (lignes 200 et 220).

SAVE

La routine **SAVE**, quant à elle, ressemble à sa sœur jumelle comme deux gouttes d'eau (cherchez les différences) :

```
690 'Routine SAVE-----------------------------------------------------
700 IF NOMBRE=0 THEN 70
710 LONGUEUR=LEN(ENTREE$):IF LONGUEUR<5 THEN 170
720 DELIMITEUR=0:FOR I=5 TO LONGUEUR
730 IF MID$(ENTREE$,I,1)=CHR$(34) THEN GOSUB 650
740 NEXT:IF DELIMITEUR=0 OR DELIMITEUR>2 THEN 170
750 IF PREMIER=LONGUEUR OR DEUXIEME-PREMIER<1 THEN 190
760 FICHIER$=MID$(ENTREE$,PREMIER+1,DEUXIEME-PREMIER-1)
770 LONGUEUR=LEN(FICHIER$)
780 DELIMITEUR=0:POSITION=0:FOR I=1 TO LONGUEUR
790 IF MID$(FICHIER$,I,1)="." THEN DELIMITEUR=DELIMITEUR+1:
    IF DELIMITEUR=1 THEN POSITION=I
800 NEXT:IF DELIMITEUR>1 OR POSITION=1 OR POSITION>LONGUEUR THEN 190
810 NOMBRE$=RIGHT$(STR$(NOMBRE),LEN(STR$(NOMBRE))-1)
820 IF DELIMITEUR=0 THEN FICHIER$=FICHIER$+"."+NOMBRE$
830 ON ERROR GOTO 210
840 OPEN FICHIER$ FOR OUTPUT AS#1
850 FOR I=1 TO NOMBRE:WRITE#1,NOM$(I),FORME$(I):NEXT
860 CLOSE#1:GOTO 220
```

Adjonction du nombre de formes au nom du fichier

Celles-ci, peu nombreuses, se rencontrent au niveau de la détermination du nom du **FICHIER** de formes auquel on rajoute, après un point, l'indication du **NOMBRE** de formes qu'il contient (lignes 810 et 820), ainsi qu'au niveau du renvoi sur erreur qui, cette fois-ci, se produit lorsque la disquette est protégée contre l'écriture (lignes 830 et 210).

INSERT

La routine suivante est celle de la commande **INSERT**. Sensiblement analogue à ses prédécesseuses, elle s'en distingue notamment par le décalage qu'elle effectue sur les **NOMS** et les **FORMES** situés au-delà du **NUMERO** de la forme à insérer (ligne 920). Notez toutefois, que l'absence de **NUMERO** à la suite de la commande **INSERT** est considérée par la ligne 890 comme la volonté d'insérer la nouvelle forme à la suite de la dernière forme du fichier. Notez également que l'incrémentation du **NOMBRE** de formes n'est exécutée qu'après que l'on ait réussi à charger le fichier de la nouvelle forme (ligne 980).

```
870 'Routine INSERT---------------------------------------------------
880 LONGUEUR=LEN(ENTREE$)
890 IF LONGUEUR<7 THEN NUMERO=NOMBRE+1:GOTO 930
900 NUMERO=VAL(RIGHT$(ENTREE$,LONGUEUR-6)):IF NUMERO=0 THEN 170
910 IF NUMERO>NOMBRE+1 THEN 180
920 FOR I=NOMBRE TO NUMERO STEP-1:
    NOM$(I+1)=NOM$(I):FORME$(I+1)=FORME$(I):NEXT
930 IF NUMERO>255 THEN 180
940 PRINT TAB(3)"Nom de fichier de la forme: ";:INPUT"",FICHIER$
950 ON ERROR GOTO 200
960 OPEN FICHIER$ FOR INPUT AS#1
970 I=NUMERO:NOM$(I)=FICHIER$:INPUT#1,FORME$(I)
980 CLOSE#1:NOMBRE=NOMBRE+1:GOTO 220
```

Déclage de gauche à droite...

DELETE

La routine **DELETE**, quant à elle, est beaucoup plus courte car elle ne nécessite aucun chargement de fichier de formes :

```
990 'Routine DELETE-----------------------------------------------------------
1000 IF NOMBRE=0 THEN 70
1010 LONGUEUR=LEN(ENTREE$):IF LONGUEUR<7 THEN 170
1020 NUMERO=VAL(RIGHT$(ENTREE$,LONGUEUR-6)):IF NUMERO=0 THEN 170
1030 IF NUMERO>NOMBRE THEN 180
1040 IF NUMERO<NOMBRE THEN FOR I=NUMERO TO NOMBRE-1:
     NOM$(I)=NOM$(I+1):FORME$(I)=FORME$(I+1):NEXT
1050 NOMBRE=NOMBRE-1
1060 GOTO 70
```

... ou de droite à gauche !

Notez que, contrairement à la routine précédente, le décalage des **NOMS** et des **FORMES** s'effectue cette fois dans le sens droite-gauche (des valeurs les plus élevées vers les valeurs les plus faibles) en débutant à partir du **NUMERO** de la forme à supprimer (ligne 1040). Quant à la décrémentation du nombre de formes, elle a lieu après exécution complète du décalage (ligne 1050).

SWAP

Ne reste plus pour finir ce gestionnaire de fichiers qu'à taper la routine **SWAP** que voici :

```
1070 'Routine SWAP-----------------------------------------------------------
1080 IF NOMBRE=0 THEN 70
1090 LONGUEUR=LEN(ENTREE$):IF LONGUEUR<5 THEN 170
1100 DELIMITEUR=0:FOR I=5 TO LONGUEUR
1110 IF MID$(ENTREE$,I,1)="-" THEN DELIMITEUR=DELIMITEUR+1:
     IF DELIMITEUR=1 THEN POSITION=I
1120 NEXT:IF DELIMITEUR=0 OR DELIMITEUR>1 THEN 170
1130 IF POSITION-5<1 OR LONGUEUR-POSITION<1 THEN 170
1140 I=VAL(MID$(ENTREE$,5,POSITION-5)):IF I=0 THEN 170
1150 IF I>NOMBRE THEN 180
1160 J=VAL(MID$(ENTREE$,POSITION+1,LONGUEUR-POSITION)):IF J=0 THEN 170
1170 IF J>NOMBRE THEN 180
1180 SWAP NOM$(I),NOM$(J):SWAP FORME$(I),FORME$(J)
1190 GOTO 70
```

La permutation a lieu ici !

Le cœur de cette routine est constitué de la ligne 1180 qui effectue la permutation des **NOMS** et des **FORMES** I et J. Le reste, c'est-à-dire les lignes 1080 et 1170 ne servent qu'à vérifier que la commande **SWAP** est bien suivie de deux numéros de formes, eux-mêmes séparés par le **DELIMITEUR** habituel qu'est le trait d'union. Un point c'est tout !

Le gestionnaire de fichiers de formes FUSFORM assure à travers 9 commandes la totalité des opérations de listage, redénomination, chargement, sauvegarde, insertion, suppression et permutation que celles-ci nécessitent.

Exercices sur le chapitre 6

1. Qu'est-ce qui permet de dessiner un trait vertical ?

- ☐ L'instruction LINE
- ☐ La lettre-code U du sous-langage DRAW
- ☐ La lettre-code D du sous-langage DRAW

2. Que réalisent les lettres-code BR4 du sous-langage DRAW ?

- ☐ Le dessin d'un trait horizontal de 4 pixels
- ☐ Un déplacement vers la droite de 4 pixels

3. Peut-on fixer la couleur d'une forme sans en dessiner le point de référence ?

- ☐ Oui, en employant la lettre-code C du sous-langage DRAW
- ☐ Non, car l'instruction PSET qui fixe la couleur dessine obligatoirement le point de référence

4. Quelles lettres-code font tourner une forme de 180 degrés ?

- ☐ A 180
- ☐ TA2

5. Que représente la forme « C1BU3RF2D2G2L2H2U2E2R »

- ☐ un astérisque
- ☐ un cercle
- ☐ un losange

6. Pourquoi vaut-il mieux grouper plusieurs formes graphiques en une bibliothèque de formes ?

- ☐ Parce que cela signifie l'écriture des programmes
- ☐ Pour accélérer la vitesse d'exécution des progammes

7. Quelle est la syntaxe d'emploi de la commande INSERT du gestionnaire de fichiers de formes FUSFORM ?

- ☐ INSERT *" nom de fichier "*
- ☐ INSERT *" numéro de fichier "*
- ☐ INSERT

CHAPITRE VII

LES CARACTÈRES GRAPHIQUES

Bien que l'IBM-PC et ses compatibles soient capables d'écrire du texte en page graphique, il n'en demeure pas moins vrai qu'ils le font de manière très incomplète puisque, pour ne citer qu'un exemple, il est impossible d'imprimer des textes au format 80 colonnes en mode graphique à moyenne résolution. C'est donc pour pallier à ces quelques insuffisances et aussi pour élargir le domaine d'emploi des caractères graphiques que ce présent chapitre leur est entièrement consacré.

Menu du chapitre 7

Affichage de caractères au format 80 colonnes.
Constitution d'une police de caractères graphiques.
Sous-programme d'écriture en page graphique.
Le retour des accents.
Comment obtenir de grosses lettres.
Les textes à lecture verticale.

Quel est votre choix : ■

AFFICHAGE DE CARACTÈRES AU FORMAT 80 COLONNES

Qu'il est dommage d'avoir dû se priver du plaisir de dessiner en couleurs les graphiques de gestion, tout ça parce que le mode graphique à moyenne résolution n'était pas capable d'afficher les textes autrement qu'au format 40 colonnes !... Je dis bien « n'était » car désormais nous allons l'obliger à le faire au format 80 colonnes et ce, grâce aux formes graphiques.

En effet, qu'est-ce qu'un caractère si ce n'est une forme graphique qui s'ignore ?... Il n'y a aucune différence apparente (tout du moins à notre niveau) entre un caractère affiché en mode graphique et une forme dessinée dans le même mode. Dans ces conditions, il ne nous reste plus qu'à mettre au point une police de caractères graphiques de largeur moitié de celles des caractères normaux et de les dessiner les uns à la suite des autres grâce à un sous-programme approprié pour obtenir l'effet désiré.

Un rapide calcul laisse entrevoir que l'on ne dispose que de 4 pixels (320/80) pour contenir un caractère graphique. Sachant qu'il faut laisser un espacement minimal d'un pixel entre caractères, cela revient à dire que l'on ne dispose que de 3 pixels de large pour dessiner un caractère. C'est peu, mais c'est faisable !... Par contre, pour la hauteur, nous disposons de 8-1 pixels, ce qui est très confortable comme hauteur (hauteur que vous pouvez réduire à 5 pixels si vous souhaitez disposer de petits caractères).

A titre d'essai, voici quelques lettres (bien connues) que l'on va afficher dans des mosaïques de 7 × 3 pixels :

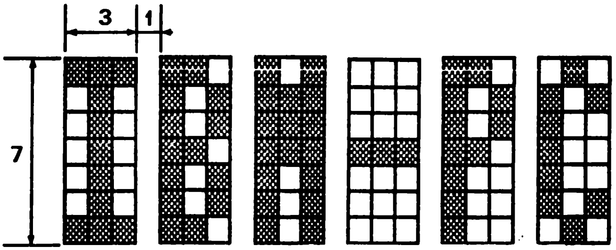


Fig. 7.1 : Quelques lettres au format 80 colonnes

Après codage des 6 formes que cette inscription nécessite, vous devriez aboutir au petit programme suivant :

Codage des 6 lettres...

```
10 KEY OFF
20 SCREEN 1:COLOR 9,1:CLS
30 I$="C=COULEUR;R2GD4GR2"
40 B$="C=COULEUR;RFBL2DBR2GLDBR2DBL2DR"
50 M$="C=COULEUR;ROBR2DL2DR2DL2DBR2DBL2DBR2DO"
60 TRAIT$="C=COULEUR;BD3R2"
70 P$="C=COULEUR;RFBL2DBR2GLD3"
80 C$="C=COULEUR;BRFBL2D4BR2G"
90 COULEUR=3
100 PRESET(148,100):DRAW I$
110 PRESET(152,100):DRAW B$
120 PRESET(156,100):DRAW M$
130 PRESET(160,100):DRAW TRAIT$
140 PRESET(164,100):DRAW P$
150 PRESET(168,100):DRAW C$
```

... et leur affichage au format 80 colonnes

Exécutez-le et observez ce qui se passe sur l'écran !... Les lettres IBM-PC s'affichent en blanc sur fond bleu au format 80 colonnes, comme si de rien n'était !... Bien sur, vous me direz que les caractères obtenus ne sont pas aussi « stylisés » qu'en mode texte. En fait, cela demeure sans grande importance dans la mesure où ils sont lisibles, ce que montre bien la copie d'écran que voici :

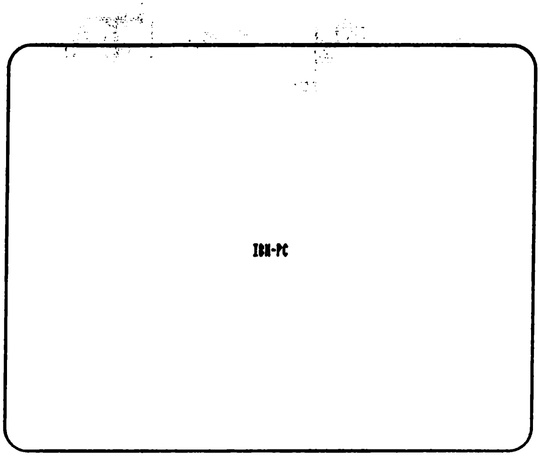


Fig. 7.2 : Affichage en couleurs de caractères graphiques au format 80 colonnes

L'affichage en couleurs de caractères graphiques au format 80 colonnes s'obtient aisément à l'aide de formes graphiques.

DÉFINITION D'UNE POLICE DE CARACTÈRES GRAPHIQUES

Le petit essai précédent étant concluant, il ne nous reste plus qu'à définir une police entière de caractères de façon à pouvoir écrire en couleur du texte au format 80 colonnes en mode graphique à moyenne résolution.

La difficulté toutefois, car c'en est une sérieuse, réside dans le volume et le temps que la saisie d'une telle police réclame. C'est pourquoi, loin de nous l'idée de vouloir incorporer les 256 caractères de la table ASCII dans cette police. Nos ambitions seront plus modestes et se limiteront aux 128 caractères (seulement...) des chiffres, majuscules, minuscules, lettres accentuées et signes de ponctuation disponibles au clavier.

Reste encore à définir le format du fichier de formes qui va contenir ces 128 caractères !... Allons-nous utiliser celui décrit au chapitre précédent ?... En fait, non, car il est tout à fait inutile d'y incorporer le nom de chaque forme, tout comme il est inutile de préciser l'ordre dans lequel elles figurent au sein du fichier.

Dans ces conditions, il est inutile également de chercher à créer le fichier de formes à l'aide du gestionnaire FUSFORM. Une bonne vieille série de DATAs devraient suffire, la preuve en est le (long...) programme que voici :

```
10 DIM POLICE$(127)
20 COULEUR=3
30 ENTETE$="C=COULEUR;"
40 FICHIER$="POLICE.128"
50 OPEN FICHIER$ FOR OUTPUT AS#1
60 FOR I=0 TO 127:READ POLICE$(I)
70 POLICE$(I)=ENTETE$+POLICE$(I)
80 WRITE#1,POLICE$(I):NEXT
90 CLOSE#1
100 DATA
110 DATA BRD4BD2R0:'!
120 DATA R0BR2DBL2DBR2D0:'"
130 DATA R0BR2DBL2DR2DBL2DR2DBL2DBR2D0:'#
140 DATA BRFLGFFGLF:'$
150 DATA BDR0BR2DGDGDBR2D0:'%
160 DATA BRFBL2DBR2GGBR2R0BGBLFR:'&
170 DATA BRDG: '
180 DATA BR2GGD2FF:'(
190 DATA R0FFD2GG:')
200 DATA BDR0BR2GGR2GGBR2D0:'*
210 DATA BFDFL2FD:'+
220 DATA BD4BFDG:',
230 DATA BD3R2:'-
240 DATA BD4BFD:'.
250 DATA BR2DGD2GD:'/
260 DATA BRFBL2DBR2DBL2DBR2DBL2F:'0
270 DATA BRDLFD3GR2:'1
280 DATA BRGBR2DGDGDR2:'2
290 DATA BRGBR2DGFDBL2F:'3
300 DATA BR2DLGBR2DBL2DR2DD:'4
310 DATA R2BGBLDRFD2BL2F:'5
320 DATA BR2LGD2RFBL2DBR2G:'6
330 DATA R2D2GD3:'7
340 DATA BRFBL2DBR2GGBR2DBL2F:'8
350 DATA BRGBR2DBL2FRD2GL:'9
360 DATA BFDBD3D:':
370 DATA BFDBD3DG:';
380 DATA BRBFGGFF:'<
390 DATA BD2R2BD3L2:'=
400 DATA BRFFGG:'>
410 DATA BRGBR2DGDBD2D0:'?
420 DATA BRFBL2DR2DL2D2FR:'@
430 DATA BRFL2DBR2DBL2DR2DBL2DBR2D0:'A
440 DATA RFBL2DBR2GLDBR2DBL2DR:'B
450 DATA BRFBL2D4BR2G:'C
460 DATA RFBL2DBR2DBL2DBR2DBL2DR:'D
470 DATA R2L2D3RGD2R2:'E
480 DATA R2L2D3RGD2:'F
490 DATA BRFBL2D3BR2DBL2FR:'G
500 DATA R0BR2DBL2DBR2DL2DBR2DBL2DBR2D0:'H
510 DATA R2GD4GR2:'I
520 DATA BR2D5BL2F:'J
530 DATA R0BR2DBL2DRDLDRFBL2DBR2D0:'K
540 DATA D6R2:'L
550 DATA R0BR2DL2DR2DL2DR2DBL2DBR2D0:'M
560 DATA R0BR2DBL2DR2DL2DR2DBL2DBR2D0:'N
570 DATA BRFBL2DBR2DBL2DBR2DBL2F:'O
580 DATA RFBL2DBR2GLD3:'P
590 DATA BRFBL2DBR2DBL2DBR2GF:'Q
600 DATA RFBL2DBR2GLDRFBL2DBR2D0:'R
610 DATA BRFBL2DFFDBL2F:'S
620 DATA R2GD5:'T
630 DATA R0BR2DBL2DBR2DBL2DBR2DBL2DR2:'U
640 DATA R0BR2DBL2DBR2DBL2DBR2DBL2F:'V
650 DATA R0BR2DBL2DBR2DL2DR2DL2DBR2D0:'W
660 DATA R0BR2DBL2FD2FBL2DBR2D0:'X
670 DATA R0BR2DBL2DBR2GD3:'Y
680 DATA R2D2GGD2R2:'Z
```

▶▶▶

▶▶▶

```
690 DATA R2L2D6R2:'[
700 DATA DFD2FD:'\
710 DATA R2D6L2:']
720 DATA BFGBR2D0:'^
730 DATA BD7R2:'_
740 DATA
750 DATA BD2RFDLGBR2DL:'a
760 DATA D3RFBL2DBR2GL:'b
770 DATA BFBDFBL2D2BR2G:'c
780 DATA BR2D3LGBR2DBL2FR:'d
790 DATA BFBDFBL2DRGFR:'e
800 DATA BRFBL2D2RGD2:'f
810 DATA BFBDFBL2DBR2DLFGL:'g
820 DATA D3RFBL2DBR2DBL2D0:'h
830 DATA BRD0BD2LFD2GR2:'i
840 DATA BR2D0BD2D3BL2DBR2G:'j
850 DATA D2BR2DBL2DRFBL2DBR2D0:'k
860 DATA RD5GR2:'l
870 DATA BD2R0BR2DL2DR2DL2DBR2D0:'m
880 DATA BD2RFBL2DBR2DBL2DBR2D0:'n
890 DATA BFBDFBL2DBR2DBL2F:'o
900 DATA BD2RFBL2DBR2GLD2:'p
910 DATA BF2LGBR2DBL2FRD2:'q
920 DATA BD2R0BR2GLD3:'r
930 DATA BF2LGF2GL:'s
940 DATA D2RGD2BR2G:'t
950 DATA BD2R0BR2DBL2DBR2DBL2FR:'u
960 DATA BD2R0BR2DBL2DBR2DBL2F:'v
970 DATA BD2R0BR2DBL2DR2DL2DBR2D0:'w
980 DATA BD2R0BR2DBL2F2BL2DBR2D0:'x
990 DATA BD2R0BR2DBL2DBR2DLFGL:'y
1000 DATA BD2R2DGGDR2:'z
1010 DATA
1020 DATA
1030 DATA
1040 DATA
1050 DATA
1060 DATA
1070 DATA BDR0BR2D0BD2L0BL2DBR2DBL2FR:'ü
1080 DATA BR2LBD2GBR2GLDFR:'é
1090 DATA R2BG2RFDLGBR2DL:'â
1100 DATA R0BR2G0BG2RFDLGBR2DL:'ä
1110 DATA BRLBD2RFDLGBR2DL:'à
1120 DATA
1130 DATA BF2LGDFRDG:'ç
1140 DATA R2BGBDGBR2GLDFR:'ê
1150 DATA R0BR2D0BDBGGBR2GLDFR:'ë
1160 DATA RBD2GBR2GLDFR:'è
1170 DATA R0BR2G0BG2RD3GR2:'ï
1180 DATA R2BG2RD3GR2:'î
1190 DATA
1200 DATA R0BR2GGBR2DBL2DR2DBL2DBR2D0:'Ä
1210 DATA
1220 DATA
1230 DATA
1240 DATA
1250 DATA BDR2BGBDGBR2DBL2F:'ô
1260 DATA BDR0BR2D0BGBDGBR2DBL2F:'ö
1270 DATA
1280 DATA BDR2BD2L0BL2DBR2DBL2FR:'û
1290 DATA
1300 DATA BDR0BR2D0BD2L0BL2DBR2DLFGL:'ÿ
1310 DATA R0BR2GGBR2DBL2DBR2DBL2F:'Ö
1320 DATA R0BR2D0BD2L0BL2DBR2DBL2DBR2DL2:'Ü
1330 DATA
1340 DATA BRFBL2D2RGD2R2:'£
1350 DATA
1360 DATA
1370 DATA
```

■■■

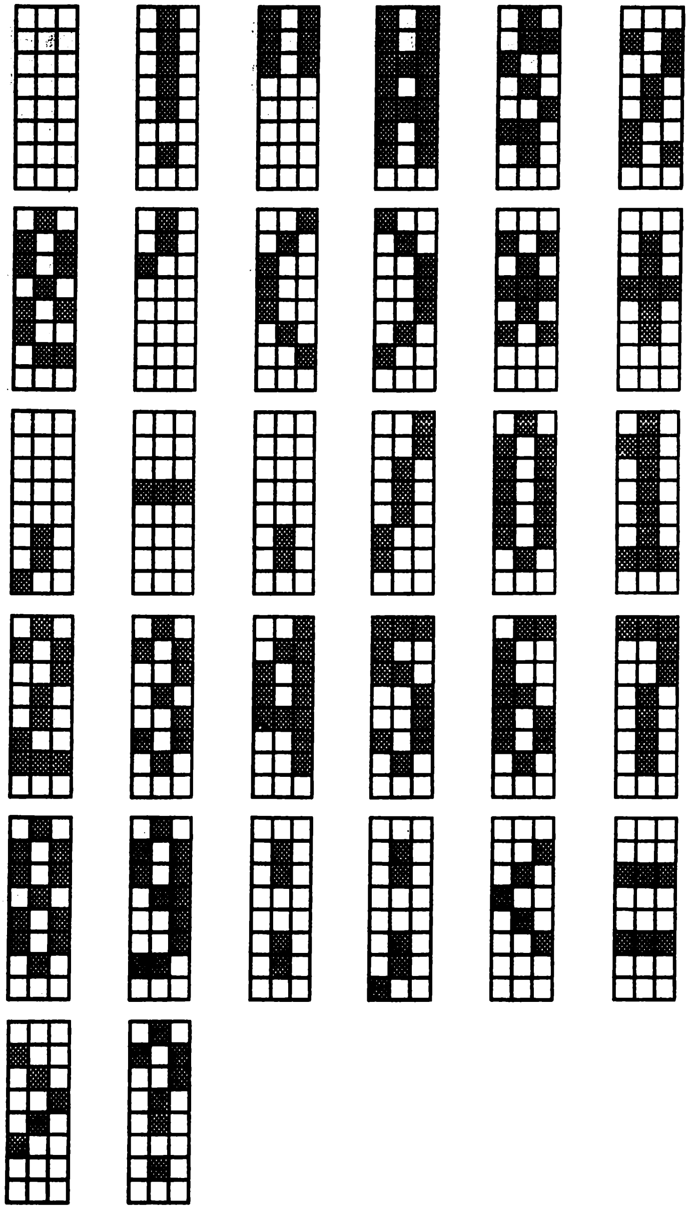


Table 7.1 : Caractères ASCII (codes 32 à 63)

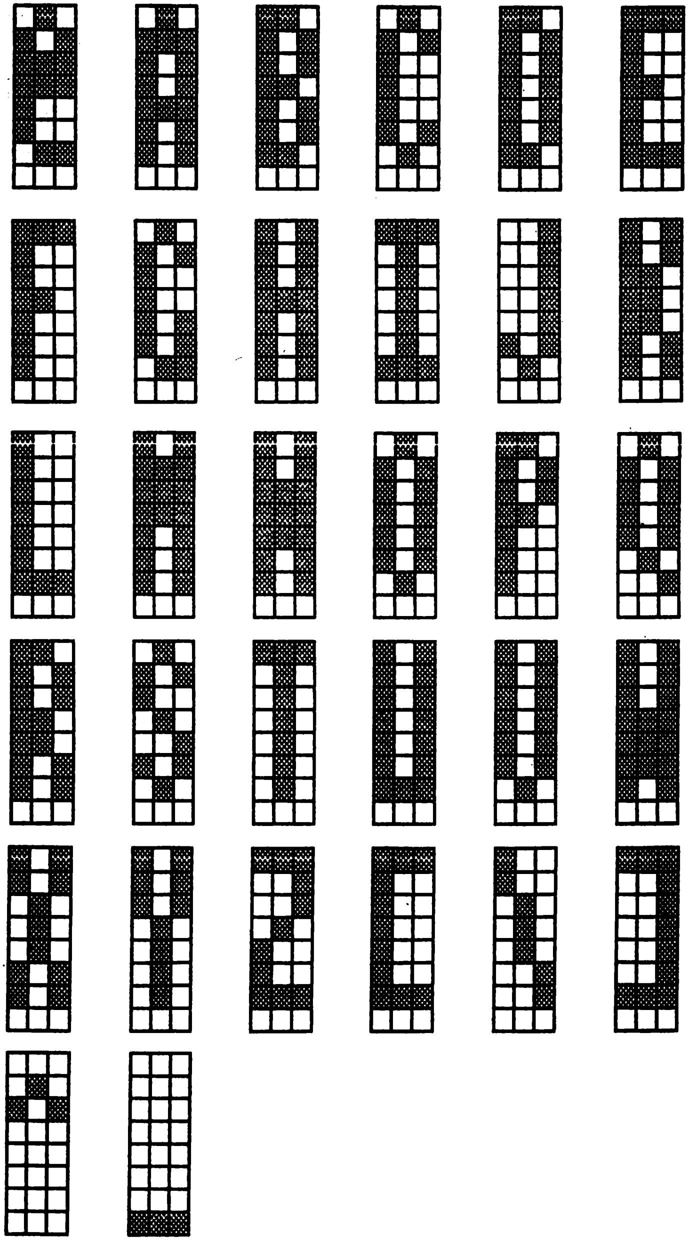
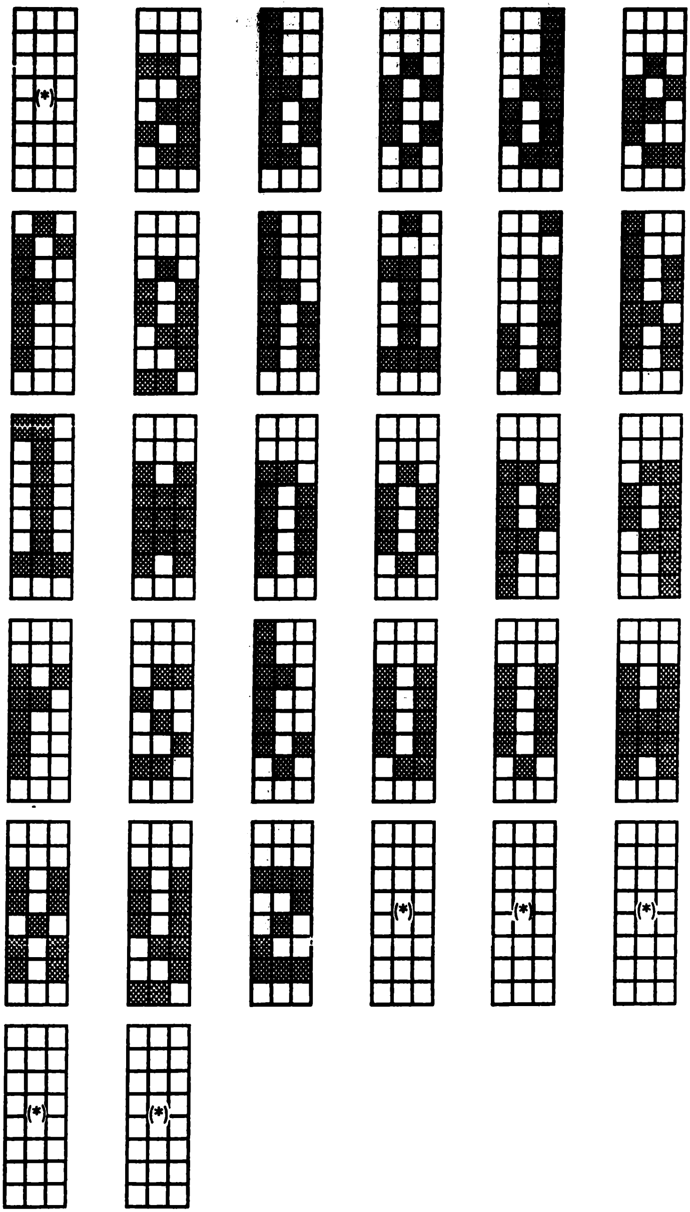
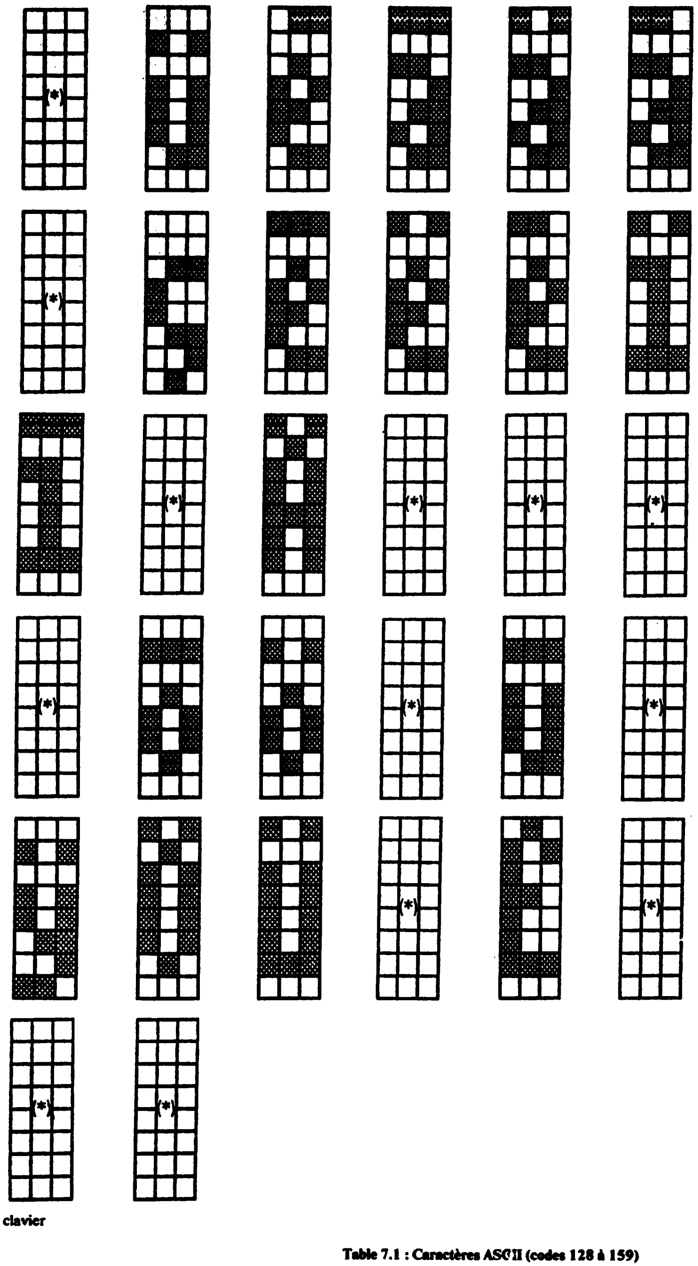


Table 7.1 : Caractères ASCII (codes 64 à 95)



(*) Non disponible au clavier

Table 7.1 : Caractères ASCII (codes 96 à 127)



(*) Non disponible au clavier

Table 7.1 : Caractères ASCII (codes 128 à 159)

Vous noterez qu'après création de la bibliothèque de formes POLICE . 128 contenant les 128 caractères graphiques, ce programme devient désormais inutile.

> **Il est plus simple de manipuler une police de caractères graphiques lorsqu'elles provient d'une bibliothèque de formes.**

SOUS-PROGRAMME D'ÉCRITURE EN PAGE GRAPHIQUE

La police de caractères étant prête, il ne reste plus qu'à créer le sous-programme qui, au même titre que l'instruction PRINT, va permettre d'écrire (en couleurs s'il vous plait !) des caractères au format 80 colonnes sur l'écran graphique à moyenne résolution :

Après chargement de la police de caractères...

```
10 DIM POLICE$(127)
20 FICHIER$="POLICE.128"
30 OPEN FICHIER$ FOR INPUT AS#1
40 FOR I=0 TO 127:INPUT#1,POLICE$(I):NEXT
50 CLOSE#1
60 KEY OFF:SCREEN 1:COLOR 9,1:CLS
70 CHAINE$="Désormais, l'écriture de textes en couleurs au format 80 colonnes su
r l'écran à moyenne résolution ne présente plus la moindre difficulté, la preuve
 !"
80 COULEUR=1:LIGNE=12:COLONNE=1:GOSUB 110
90 END
100 'Sous-programme PRINT-----------------------------------------------
110 IF LIGNE=0 THEN LIGNE=1
120 IF COLONNE=0 THEN COLONNE=1
130 LONGUEUR=LEN(CHAINE$):IF LONGUEUR=0 THEN 180
140 FOR I=1 TO LONGUEUR:ASCII=ASC(MID$(CHAINE$,I,1))
150 X=(COLONNE-1)*4:Y=(LIGNE-1)*8
160 PRESET (X,Y):DRAW POLICE$(ASCII-32)
170 IF ASCII<32 OR ASCII>159 THEN ASCII=32
180 COLONNE=COLONNE+1:IF COLONNE>80 THEN COLONNE=1:
    IF LIGNE<25 THEN LIGNE=LIGNE+1
190 NEXT
200 IF DELIMITEUR$="" THEN COLONNE=1:
    IF LIGNE<25 THEN LIGNE=LIGNE+1
210 DELIMITEUR$="":RETURN
```

... ce sous-programme écrit en page graphique comme si de rien n'était !

Vous noterez que ce sous-programme est rattaché à un petit programme de démonstration (lignes 10 à 90) chargé d'afficher en turquoise sur fond bleu le texte que voici :

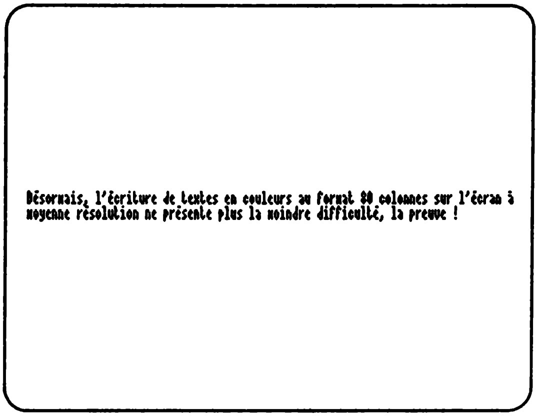


Fig. 7.3 : Exemple d'écriture en moyenne résolution (format 80 colonnes)

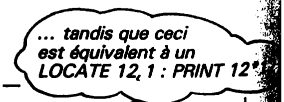
Voyons un peu comment tout cela fonctionne !... La ligne 80 du programme de démonstration appelle le sous-programme **PRINT**, non sans avoir au préalable fixé la valeur de la **LIGNE** et de la **COLONNE** de tabulation (c'est l'équivalent d'un **LOCATE**).

Ceci fait, on aborde le sous-programme (lignes 110 et 120) en prenant moults précautions, puisque l'on s'assure de la non nullité des valeurs de tabulation, phénomène qui peut se produire lorsqu'on appelle pour la première fois sans **LOCATE** préalable le sous-programme **PRINT**, comme dans l'exemple suivant :

```
70 CHAINE$="Bonjour"
80 GOSUB 110
```

Vous remarquerez que ceci est équivalent à l'emploi de l'instruction **PRINT** sans **LOCATE** préalable et se traduit par l'affichage du texte de la **CHAINE$** au début de la 1ère ligne lors du premier appel du sous-programme **PRINT** et au début des lignes suivantes lors des appels ultérieurs.

Puisque l'on est dans les équivalences, voici ce qu'il vous faut écrire pour obtenir l'affichage d'une expression numérique. En l'occurence, il s'agit du produit de 12 × 12 :




```
70 CHAINE$=STR$(12*12)
80 LIGNE=12:COLONNE=1:GOSUB 110
```

Dans le même ordre d'idées, voici comment il vous faut introduire le DELIMITEUR qu'est le point-virgule pour enchaîner les affichages les uns à la suite des autres :

Notez la présence du point-virgule qui évite le retour à la ligne

```
70 LIGNE=12:COLONNE=1:CHAINE$="12+12=":DELIMITEUR$=";":GOSUB 110
80 CHAINE$=STR$(12+12):GOSUB 110
```

Ceci dit, revenons à l'analyse du sous-programme **PRINT**. La ligne 130 mesure la **LONGUEUR** de la **CHAINE** et, si celle-ci est nulle, effectue le branchement vers la ligne 200 de test du retour ligne.

La **LONGUEUR** de la **CHAINE** n'étant pas nulle, les lignes 140 à 190 en effectuent la lecture caractère par caractère en leur attribuant au passage leur code **ASCII** (ligne 140), code **ASCII** que l'on filtre (ligne 150) de façon à ne pas déborder des 128 caractères de la **POLICE** chargée en début de programme. Ce filtrage étant assuré, les lignes 160 et 170 procèdent à l'affichage des formes graphiques représentant les caractères et la ligne 180 assure l'incrémentation du numéro de colonne de manière à afficher les caractères les uns à la suite des autres. Quant à la ligne 200, elle exécute systématiquement un retour à la ligne en absence de **DELIMITEUR**.

Vous pensiez en avoir terminé avec le sous-programme **PRINT** ?... Eh bien non !... Sachez que non seulement content d'afficher des textes en moyenne résolution au format 80 colonnes, il est capable de faire deux fois mieux en haute résolution, c'est-à-dire qu'il peut afficher des textes au format 160 colonnes.

Or, point n'est besoin de grandes transformations pour parvenir à un tel résultat !... Il vous suffit de modifier comme suit les lignes 60 et 180...

Passage en mode graphique à haute résolution...

```
60 KEY OFF:SCREEN 2:CLS:COULEUR=1
180 COLONNE=COLONNE+1:IF COLONNE>160 THEN COLONNE=1:
     IF LIGNE<25 THEN LIGNE=LIGNE+1
```

... et limitation du nombre de caractères à 160 par ligne !

... pour aboutir au résultat que voilà :

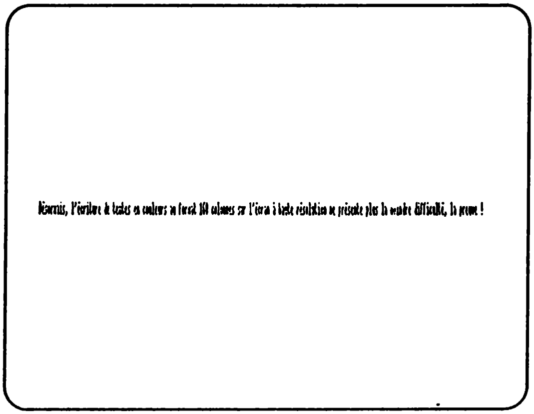


Fig. 7.4. Exemple d'écriture en haute résolution (format 160 colonnes)

Le sous-programme PRINT permet d'afficher des textes au format 80 colonnes en moyenne résolution ou au format 160 colonnes en haute résolution.

LE RETOUR DES ACCENTS

L'exercice précédent, n'était destiné qu'à permettre l'écriture de textes dans des formats non prévus par le constructeur. Or une telle démarche, pour intéressante qu'elle soit, peut très bien ne pas retenir votre attention si rien ne vous pousse à vouloir employer d'autres formats que ceux existants.
Pourtant, il est un point que tout amoureux de la langue française ne peut accepter, j'ai nommé l'absence de lettres accentuées en mode graphique !... Vous n'aviez pas remarqué ?... Alors, essayez vite le petit programme suivant...

```
10 KEY OFF:SCREEN 1:COLOR 9,1:CLS
20 LOCATE 12,1:PRINT"Les lettres accentuées ont été oubliées en mode graphique."
```

... qui produit le résultat (inattendu) que voilà :

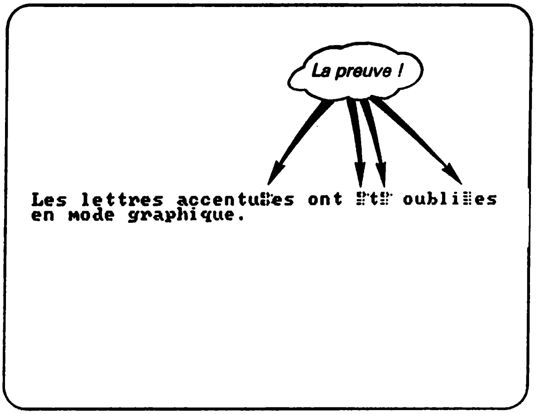
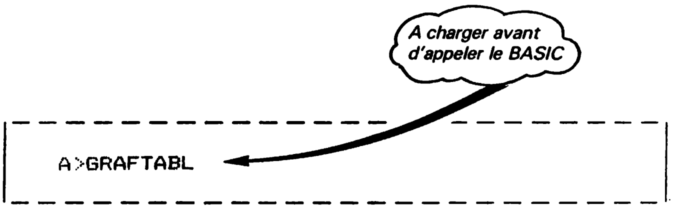


Fig. 7.5 : Tentative (infructueuse) d'affichage des lettres accentuées en mode graphique.

Est-ce à dire qu'il va falloir se résigner à n'utiliser que des lettres majuscules en mode graphique ?... Non, bien sûr, puisqu'il vous suffit de charger, sous DOS, la police complémentaire de caractères dénommée GRAFTABL comportant (entre autre...) les 32 **caractères manquants qui correspondent aux lettres accentuées** (codes ASCII 128 à 159) :



Une fois que vous avez chargé cette police complémentaire de caractères graphiques il ne vous reste plus qu'à vérifier que les lettres accentuées s'obtiennent désormais aussi facilement en mode graphique qu'en mode texte. Pour ce faire, il vous suffit de taper :

```
10 KEY OFF:SCREEN 1:COLOR 9,1:CLS
20 LOCATE 12,1:PRINT "     Grâce à la police de caractères"
30 LOCATE 13,1:PRINT "    graphiques GRAFTABL, les lettres"
40 LOCATE 14,1:PRINT "accentuées s'obtiennent aussi facilement"
50 LOCATE 15,1:PRINT "   en mode graphique qu'en mode texte"
```

L'exécution de ce court programme vous montrera que, grâce au chargement préalable de GRAFTABL, la situation est redevenue normale :

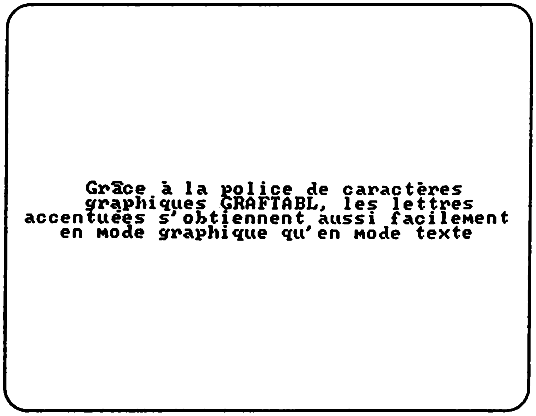


Fig. 7.6 : Affichage (réussi) de caractères accentués en mode graphique.

L'affichage des caractères accentués en mode graphique nécessite le chargement préalable de la police de caractères graphiques GRAFTABL.

COMMENT OBTENIR DE GROSSES LETTRES

L'affichage des caractères accentués étant désormais chose acquise, nous allons nous orienter maintenant vers un autre exercice, à savoir l'obtention de grosses lettres à l'échelle 2, 4 ou 8.

Pourquoi des grosses lettres vous demandez-vous ?... Mais pour réaliser un titeur, c'est-à-dire un utilitaire qui va vous servir à réaliser des titres en couleurs (pour être photographiés) ou en noir et blanc (pour être imprimés). Si donc une telle démarche vous intéresse, suivez le guide !...

POINT

Voyons, que faut-il pour obtenir de grosses lettres sur l'écran ?... Rien de bien spécial, si ce n'est la police graphique complémentaire GRAFTABL chargée précédemment et l'instruction **POINT**(*). Pourquoi l'instruction **POINT** ?... Parce que nous allons dessiner les caractères un à un dans un coin de l'écran et nous allons les analyser pixel par pixel avec cette instruction qui, comme son nom le laisse supposer, délivre la couleur du pixel adressé. A titre d'exemple, POINT (160, 100) est égal à 0 lorsque le pixel des coordonnées graphiques 160, 100 est dessiné dans la couleur de fond de l'écran et il est égal à 2 lorsque celui-ci est de couleur rouge (palette n° 0) ou violette (palette n° 1).

Mais, trêve d'explications, passons au programme (en fait c'est un sous-programme) qui permet de dessiner des caractères à l'échelle 2, 4 ou 8 selon le choix :

```
10 KEY OFF:SCREEN 1:COLOR 9,1:CLS:COULEUR=3
20 CHAINE$="LES":ECHELLE=8:LIGNE=3:COLONNE=9:GOSUB 70
30 CHAINE$="GROSSES":ECHELLE=4:LIGNE=13:COLONNE=7:GOSUB 70
40 CHAINE$="LETTRES":ECHELLE=2:LIGNE=19:COLONNE=14:GOSUB 70
50 END
60 'Sous-programme d'affichage de grosses lettres---------------
70 IF LIGNE=0 THEN LIGNE=1
80 IF COLONNE=0 THEN COLONNE=1
90 LONGUEUR=LEN(CHAINE$):IF LONGUEUR=0 THEN 180
100 FOR I=1 TO LONGUEUR:ASCII=ASC(MID$(CHAINE$,I,1))
110 LOCATE 25,1:PRINT CHR$(ASCII);
120 FOR V=0 TO 7:FOR U=0 TO 7
130 X=(COLONNE-1)*8+U*ECHELLE:Y=(LIGNE-1)*8+V*ECHELLE
140 IF POINT(U,192+V)<>0 THEN
    LINE(X,Y)-(X+ECHELLE-1,Y+ECHELLE-1),COULEUR,BF
150 NEXT U,V:LOCATE 25,1:PRINT" ";
160 COLONNE=COLONNE+ECHELLE:IF COLONNE>40 THEN COLONNE=1:
    IF LIGNE<25 THEN LIGNE=LIGNE+ECHELLE
170 NEXT
180 IF DELIMITEUR$="" THEN COLONNE=1:
    IF LIGNE<25 THEN LIGNE=LIGNE+ECHELLE
190 DELIMITEUR$="":RETURN
```

Echelle de reproduction (renvoi vers ECHELLE=2 de la ligne 40)

(*) La fiche détaillée de l'instruction POINT figure à la page 329 du volume 2 de Pratique des IBM-PC et compatibles.

Que pensez-vous du résultat obtenu ?... Pas mal, n'est-ce pas !...

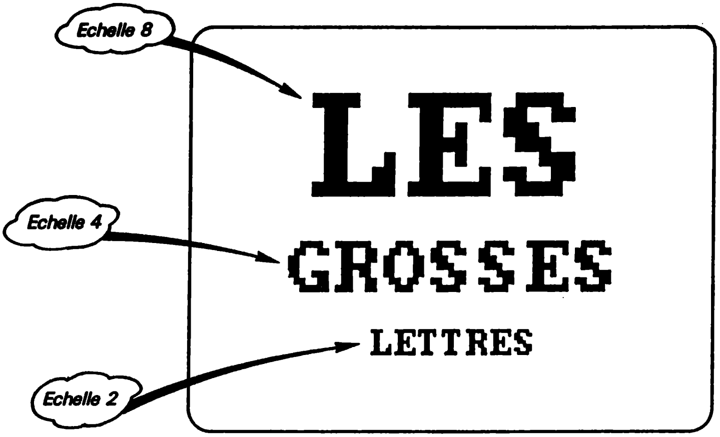


Fig. 7.7 : Affichage de caractères à l'échelle 2, 4 et 8.

Côté fonctionnement, ce sous-programme ne diffère du précédent que par les lignes 110 à 160, le reste étant strictement identique. Tout d'abord, la ligne 110 se charge d'afficher chaque caractère à tour de rôle dans l'angle inférieur gauche, qu'il s'agisse d'un caractère normal ou d'une lettre accentuée. Ceci fait, les lignes 120 à 140 analysent pixel par pixel (il y en a 8 × 8) le caractère ainsi affiché afin de repérer la couleur de chacun d'entre-eux. Si celle-ci diffère de la couleur de fond, le pixel est alors dessiné à l'ECHELLE souhaitée (2, 4 ou 8). Notez la méthode employée pour dessiner des carrés de taille croissante, méthode qui fait appel aux possibilités de dessin de boîtes pleines de l'instuction LINE (ligne 140).

Reste la ligne 150 à expliquer !... A quoi sert-elle ?... A première vue, elle ne sert à rien puisqu'elle efface (en dessinant un espace par dessus) le caractère témoin avant d'afficher le suivant, opération qui efface une seconde fois le caractère précédent. Or, cela est indispensable si l'on veut éviter, de laisser traîner derrière soi le dernier caractère témoin affiché.

Remarque : *Bien que ce sous-programme d'écriture de grosses lettres ait été prévu pour les valeurs d'ECHELLE 2, 4 et 8, rien ne vous empêche d'employer d'autres valeurs.*

L'affichage de grosses lettres s'obtient aisément avec les polices de caractères existantes, qu'il s'agisse des lettres normales ou des lettres accentuées.

LES TEXTES À LECTURE VERTICALE

La méthode utilisée pour l'affichage des grosses lettres et à ce point puissante qu'elle peut être employée à toute une foule d'applications comme par exemple l'effet miroir (qui consiste à écrire un texte lisible seulement dans un miroir), le dessin d'ombres à l'arrière des caractères (cela en accentue le relief), leur inclinaison par rapport à l'axe horizontal, etc, etc...

Comme la place nous manque pour traiter chaque cas, nous allons terminer ce chapitre avec les caractères à lecture verticale, tels que ceux que l'on peut voir sur le diagramme de gestion ci-après :

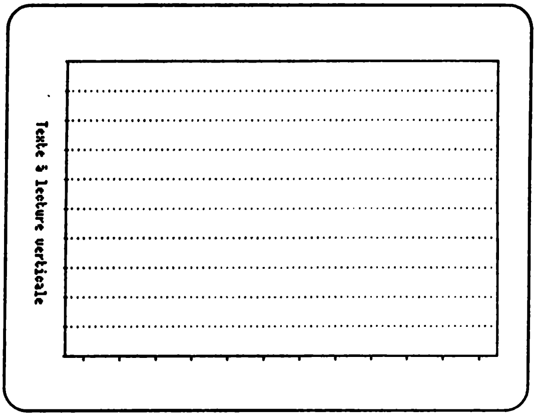


Fig. 7.8 : Exemple de texte à lecture verticale.

Fort utiles pour désigner l'axe des ordonnées des diagrammes à repères orthogonaux, les textes à lecture verticale ne sont réellement exploitables qu'en mode graphique à haute résolution car l'on dispose alors de colonnes de 50 caractères... à condition bien entendu que chaque caractère s'inscrive dans une mosaïque de 16 × 4 points.

Pourquoi 16 × 4 points ?... Mais pour conserver les proportions des caractères lorsqu'ils sont dessinés sur le côté, la résolution horizontale du mode à haute résolution étant le double de sa résolution verticale !... C'est pourquoi vous allez retrouver dans le petit programme (et sous-programme) de démonstration que voici, la bonne vieille police de caractères étroits définie au début de ce chapitre :

Police de 128 caractères étroits

```
10 DIM POLICE$(127)
20 FICHIER$="POLICE.128":OPEN FICHIER$ FOR INPUT AS#1
30 FOR I=0 TO 127:INPUT#1,POLICE$(I):NEXT:CLOSE#1
40 KEY OFF:SCREEN 2:CLS:COULEUR=1
50 LINE(58,11)-(634,171),,B:LINE(59,11)-(635,171),,B
60 FOR I=1 TO 12:X=58+(I-.5)*48:
   LINE(X,171)-(X,173):LINE(X+1,171)-(X+1,173):NEXT
70 FOR Y=11 TO 171 STEP 16:LINE(56,Y)-(61,Y):
   LINE(64,Y)-(631,Y),,,&H404:NEXT
80 CHAINE$="Texte à lecture verticale":LIGNE=12:COLONNE=3:GOSUB 110
90 END
```

```
100 'Sous-programme d'affichage de textes à lecture verticale------------------
110 IF LIGNE=0 THEN LIGNE=1
120 IF COLONNE=0 THEN COLONNE=1
130 LONGUEUR=LEN(CHAINE$):IF LONGUEUR=0 THEN 230
140 FOR I=1 TO LONGUEUR:ASCII=ASC(MID$(CHAINE$,I,1))
150 IF ASCII<32 OR ASCII>159 THEN ASCII=32
160 PRESET(0,192):DRAW POLICE$(ASCII-32)
170 FOR V=0 TO 7:FOR U=0 TO 2
180 X=(COLONNE-1)*16-V*2:Y=(LIGNE-1)*4+U
190 IF POINT(U,192+V)<>0 THEN PSET(X,Y):PSET(X-1,Y)
200 NEXT U,V:LOCATE 25,1:PRINT" ";
210 LIGNE=LIGNE+1:IF LIGNE>50 THEN LIGNE=1:
    IF COLONNE>0 THEN COLONNE=COLONNE-1
220 NEXT
230 IF DELIMITEUR$="" THEN LIGNE=1:
    IF COLONNE>0 THEN COLONNE=COLONNE-1
240 DELIMITEUR$="":RETURN
```

Cherchez les différences par rapport au sous-programme précédent !... La première réside à la ligne 150 où s'effectue le test du code **ASCII** du caractère. La deuxième, vous l'avez à la ligne 180 de calcul des coordonnées X et Y, le pas horizontal étant de 16 pixels par caractère alors que le pas vertical n'est que de 4 pixels par caractère. Quant à la troisième, elle est à la ligne 190 qui répète sur 2 pixels contigus le même motif de façon à doubler l'épaisseur des traits.

Est-ce bien tout ?... Non, car il reste à inverser l'incrémentation des lignes et des colonnes, les lignes devenant les colonnes et vice-versa (lignes 210 et 230).

Remarque : *Il peut paraître absurde de dessiner une forme graphique dans un coin de l'écran pour ensuite la reproduire inclinée sur le côté, alors qu'il existe une lettre-code dans le sous-langage DRAW qui effectue directement cette rotation. La raison en est simple !... C'est le seul moyen qui permette de l'agrandir 2 fois selon l'axe horizontal et une fois seulement selon l'axe vertical.*

L'affichage en haute résolution de textes à lecture verticale nécessite l'utilisation de la police de 128 caractères étroits définie au début de ce chapitre.

Exercices sur le chapitre 7

1. Les caractères graphiques résidants s'inscrivent dans une mosaïque de :

☐ 8 × 4 pixels
☐ 8 × 7 pixels
☐ 8 × 8 pixels

2. A quoi cela peut-il bien servir de définir une police de 128 caractères graphiques ?

☐ à écrire du texte au format 80 colonnes en moyenne résolution
☐ à écrire du texte au format 160 colonnes en haute résolution
☐ à écrire du texte à lecture verticale à haute résolution

3. Comment obtient-on les caractères accentués en mode graphique ?

☐ en procédant comme en mode texte
☐ en chargeant la police complémentaire de caractères graphiques dénommée GRAFTABL.

4. Que réalise l'instruction POINT ?

☐ elle indique la couleur d'un pixel allumé
☐ elle précise si un point est allumé ou éteint

5. Comment peut-on doubler la largeur des caractères ?

☐ en employant une police de caractères élargis
☐ en doublant la largeur de chaque pixel

6. Comment peut-on diviser par deux la largeur des caractères ?

☐ en employant une police de caractères étroits
☐ en divisant par deux la largeur de chaque pixel.

Réponses page 154.

Solution des exercices sur le chapitre 6

1. L'instruction LINE et la lettre-code U et la lettre-code D du sous-langage DRAW.
2. Un déplacement vers la droite de 4 pixels.
3. Oui, en employant la lettre-code C du sous-langage DRAW.
4. Aucune des deux !...
5. Un cercle.
6. Parce que cela signifie l'écriture des programmes
7. INSERT *numéro de forme* et INSERT.

CHAPITRE VIII

L'ANIMATION GRAPHIQUE

Et si l'on animait toutes ces images un peu trop figées ?... Que diriez-vous, si l'on vous montrait commen déplacer une main sur l'écran, faire rebondir une balle sur un mur, décoller et atterir un hélicoptère, marcher un personnage, rouler une voiture, etc, etc ?... Bref, en un mot, êtes-vous intéressé par l'animation graphique ?... Si c'est le cas, alors plongez à fond dans ce chapitre qui est entièrement consacré à ce sujet passionnant.

Menu du chapitre 8

Comment créer le mouvement.
Conservation du décor de fond.
Les instructions GET et PUT
Le contrôle des déplacements à partir du clavier.
Séquence d'animation nécessitant plusieurs formes graphiques
Changements alternés de plans d'animation
Entrée et sortie progressive de l'écran.

Quel est votre choix : ■

COMMENT CRÉER LE MOUVEMENT

Les formes graphiques n'ayant désormais plus de secret pour vous, il vous appartient de leur apprendre à se mouvoir au même titre qu'on apprend à marcher à un jeune enfant... Toutefois, la question se pose de savoir sur quelle forme graphique vous allez jeter votre dévolu ?... Quoique rien ne vous empêche de choisir un modèle assez élaboré, il semble plus raisonnable de faire dans le simple (tout du moins à ce stade de l'ouvrage).

C'est pourquoi il vous est proposé, en tant que permier exemple, d'animer le rebond d'une balle sur les bords de l'écran. Vous y apprendrez (entre autre...) l'écriture qui permet de créer le mouvement d'un objet ou d'un personnage. En l'occurence, il s'agit de celui d'une balle de tennis :

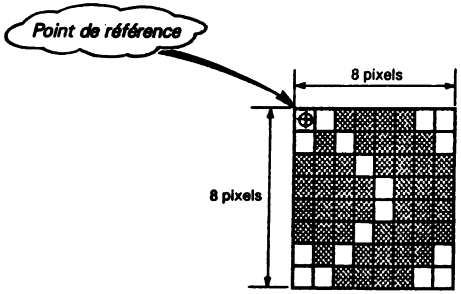


Fig. 8.1 : Forme graphique représentant une balle de tennis

Or, que vous faut-il pour parvenir à un tel résultat ?... Bien peu de choses en vérité, preuve en est le (très court) programme que voici :

```
10 KEY OFF:SCREEN 1:COLOR 8,0:CLS
20 BALLE$="C=COULEUR;BR2R3FL3BL2GR2BR2R3DL2BL2L3DR3BR2R2DL3BL2L2FBR2R3GL3"
30 LINE(3,3)-(316,196),2,B:PAINT(0,0),2
40 X=4:Y=4:H=-8:V=-8
50 COULEUR=3:PRESET(X,Y):DRAW BALLE$
60 IF X=4 OR X=308 THEN H=-H
70 IF Y=4 OR Y=188 THEN V=-V
80 COULEUR=0:PRESET(X,Y):DRAW BALLE$
90 X=X+H:Y=Y+V
100 GOTO 50
```

Tapez-le et vérifiez que vous obtenez bien une balle qui rebondit inlassablement sur les 4 bords de l'écran :

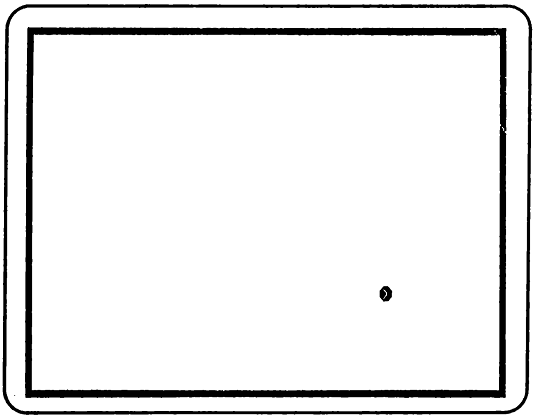


Fig. 8.2 : Rebond d'une balle sur les bords de l'écran

Ceci étant, voyons un peu comment tout cela fonctionne !... Tout d'abord, vous avez reconnu à la ligne 20 le codage de la forme graphique de la **BALLE**, suivi à la ligne 30 du dessin du pourtour de l'écran. Puis, c'est au tour de la ligne 40 d'affecter les valeurs initiales aux variables X et Y de positionnement de la balle, ainsi qu'aux variables H et V d'incrémentation des déplacements horizontaux et verticaux.

Ces 4 premières lignes constituent ce que l'on pourrait appeler la partie statique du programme. La partie dynamique, c'est-à-dire celle qui assure le mouvement de la balle, commence à partir de la ligne 50. Cette partie renferme (toujours) les 4 fonctions importantes suivantes :

- le dessin de la balle (ligne 50)
- le test des limites de déplacements (lignes 60 et 70).
- l'effacement de la balle (ligne 80).
- l'incrémentation (ou la décrémentation) des variables de position de la balle (ligne 90).

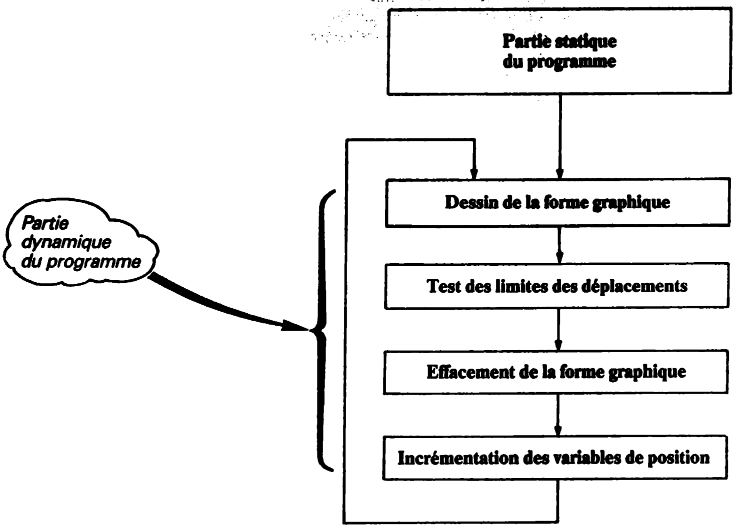


Fig.8.3 : Organigramme du mouvement

Vous noterez dans le présent organigramme, que la partie dynamique du programme commence par le dessin de la forme graphique (la balle dans notre exemple). Or, cette disposition, pour logique qu'elle soit, n'est valable que dans la mesure où l'animation ne porte que sur une seule forme graphique. Dans le cas où le programme en manipule plusieurs, il vous faut débuter la partie dynamique du programme par l'effacement de la première forme graphique. Bien entendu, cela sous-entend que cette forme (et les autres également) ait été préalablement dessinée dans la partie statique du programme.

Un autre point important de ce petit programme d'animation concerne le clignotement dont est affectée la balle. A quoi cela peut-il bien être dû ?... Mais au fait qu'après avoir effacé la balle (ligne 80), on ne la redessine que trois lignes plus loin (c'est-à-dire à la ligne 50), ce qui crée un court intervalle de temps pendant lequel rien n'est affiché sur l'écran. On a donc intérêt, si l'on cherche à minimiser autant que faire ce peut ce phénomène de clignotement, à placer la ligne de dessin de la forme graphique au plus près de celle qui l'efface. Toute l'astuce et là !

Le mouvement d'une forme graphique nécessite
les 4 fonctions élémentaires que sont :
- le dessin de la forme graphique
- le test des limites des déplacements
- l'effacement de la forme
- l'incrémentation des variables de position

CONSERVATION DU DÉCOR DE FOND

A voir la balle ainsi rebondir d'un bord à l'autre, on en vient au bout de quelques minutes de réflexion à se demander ce qu'il advient du décor de fond sur lequel elle se déplace. Est-il effacé en même temps que la balle ou bien est-il conservé ?..

Pour répondre à une telle question, rien ne vaut de vérifier la chose par soi-même !... En l'occurence, rajoutez donc le filet central que voici et observez ce qui se passe :

```
35 LINE(158,4)-(158,195),1:LINE(161,4)-(161,195),1
```

Filet dans lequel...

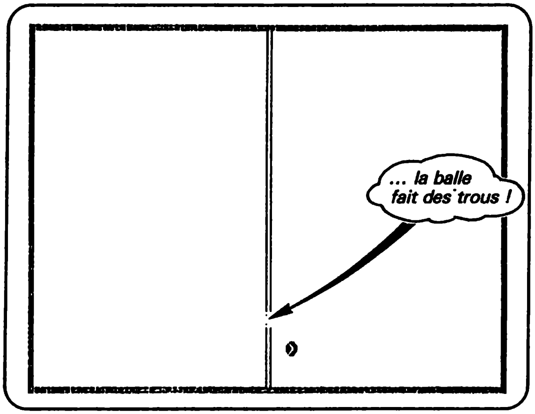


Fig. 8.4 : La balle fait des trous dans le filet

Or, que constatez-vous ?... Que la vitesse de la balle est telle qu'elle en déchire le filet !... Dur, dur !... C'est donc la preuve que les dessins et effacements successifs d'une forme graphique en mouvement transforment le décor de fond en un gruyère constellé de trous (c'est le meilleur, paraît-il !).

Est-ce sans solution ?... Non, bien entendu !... Il suffit pour résoudre ce « léger » problème de disposer d'une instruction unique qui dessine chaque pixel de la forme graphique dans la couleur complémentaire du pixel du fond qui lui correspond. Ainsi, après deux emplois successifs de la même instruction (ce qui correspond à un dessin et à un effacement) on retrouve immanquablement le décor de fond dans son intégralité originelle.

La difficulté, c'est qu'il n'existe pas dans le Basic Microsoft de l'IBM-PC d'instruction (qui pourrait s'appeler **XDRAW** par exemple) capable de dessiner une forme graphique dans la couleur complémentaire du fond (pour une prochaine version peut-être...).

Ce qui existe par contre, c'est un jeu d'instructions dénommées **GET** et **PUT** qui, spécialisées dans le traitement des images, assurent pour l'une d'entre-elles (**PUT** en l'occurence) le dessin des pixels d'une image dans la couleur complémentaire des pixels du décor de fond.

Si donc l'on cherche à conserver le décor de fond sur lequel on déplace une forme, il ne reste plus qu'à relever les manches et à aborder l'étude des instructions **GET** et **PUT**. Au passage, on découvrira qu'elles sont infiniment plus puissantes que le sous-langage graphique **DRAW** sur lequel toutefois, elles s'appuient pour constituer l'image d'une forme graphique.

LES INSTRUCTIONS GET ET PUT

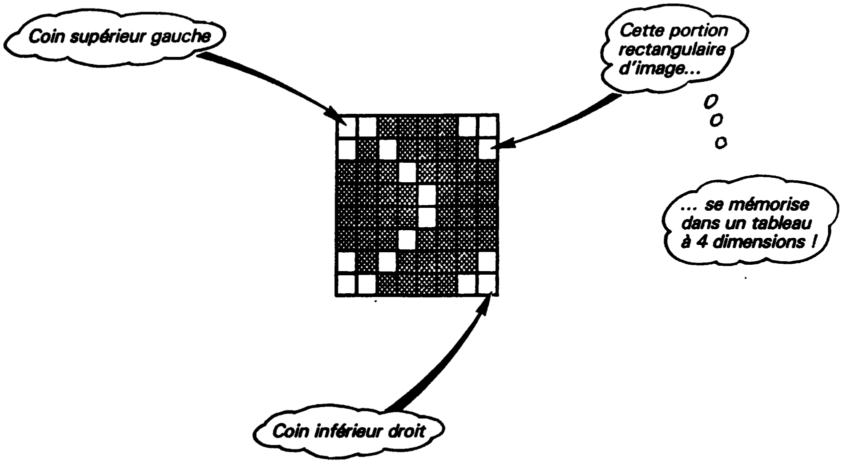
GET

Que font donc ces fameuses instructions **GET** et **PUT** ?... La première des deux, c'est-à-dire **GET**, transfère une portion rectangulaire d'image dans un tableau. De ce fait, la syntaxe exacte de cette instruction ressemble à :

GET *(coordonnées coin supérieur gauche) - (coordonnées coin inférieur droit), nom de tableau.*

Or, qui dit tableau dit dimensionnement du dit tableau (à répéter rapidement dix fois de suite) et, ce d'autant plus que l'instruction **GET n'accepte pas** le dimensionnement par défaut que réalise le Basic Microsoft en d'autres circonstances.

Dans ces conditions, comment fait-on pour déterminer la taille optimale du tableau ?... On applique une formule simplifiée qui consiste à diviser par 16 (mode graphique à moyenne résolution) ou par 32 (mode à haute résolution) le nombre de pixels contenus dans la zone rectangulaire que vous cherchez à conserver, à condition, toutefois, que le nombre de pixels de la largeur et de la longueur de la zone rectangulaire soit un multiple de 4. Si ce n'est pas le cas, il vous faut appliquer la formule exacte qui figure sur la fiche détaillée de l'instruction GET (*).



Ainsi, dans l'exemple de la balle qui est le nôtre, le nombre de pixels de la zone rectangulaire (disons plutôt carrée...) d'image est de 8 × 8 = 64, ce qui nécessite un tableau à 64/16 = 4 indices si l'on est en moyenne résolution.

(*) La fiche détaillée de l'instruction GET figure à la page 316 du volume 2 de Pratique des IBM-PC et compatibles.

PUT

Au tour maintenant de l'instruction PUT (*) qui, vous vous en doutez, va afficher sur l'écran le contenu du tableau et ce, dans la couleur complémentaire du fond. Pour cela, il suffit de lui préciser les coordonnées du coin supérieur gauche de l'image mémoriésée par GET ainsi que le nom du tableau :

PUT (*coordonnées coin supérieur gauche*), *nom de tableau.*

Il vous faut, toutefois, prendre garde en définissant les coordonnées du coin supérieur gauche, à ce qu'aucune portion de l'image mémorisée ne sorte de l'écran. Or, cela se produit d'autant plus facilement que rien dans le tableau ne précise la hauteur ni la largeur de l'image qu'il contient. Donc, gare, sinon c'est le « plantage » garanti.

Ceci dit, il est temps de revenir à votre programme de rebond d'un balle afin d'y incorporer les instructions GET et PUT qui vont empêcher la balle de transformer le filet en gruyère. Une fois modifié, le programme ressemble à ceci :

```
10 DIM BALLE(4)
20 KEY OFF:SCREEN 1:COLOR 8,0:CLS
30 BALLE$="C=COULEUR;BR2R3FL3BL2GR2BR2R3DL2BL2L3DR3BR2R2DL3BL2L2FBR2R3GL3"
40 LINE(3,3)-(316,196),2,B:PAINT(0,0),2
50 LINE(158,4)-(158,195),1:LINE(161,4)-(161,195),1
60 X=4:Y=4:H=8:V=8
70 COULEUR=3:PRESET(X,Y):DRAW BALLE$
80 GET(X,Y)-(X+7,Y+7),BALLE
90 PUT(X,Y),BALLE
100 X=X+H:Y=Y+V
110 PUT(X,Y),BALLE
120 IF X=4 OR X=308 THEN H=-H
130 IF Y=4 OR Y=188 THEN V=-V
140 GOTO 90
```

Essayez-le !... Vous verrez, le résultat est concluant puisque le filet demeure intact malgré la danse incessante de la balle.

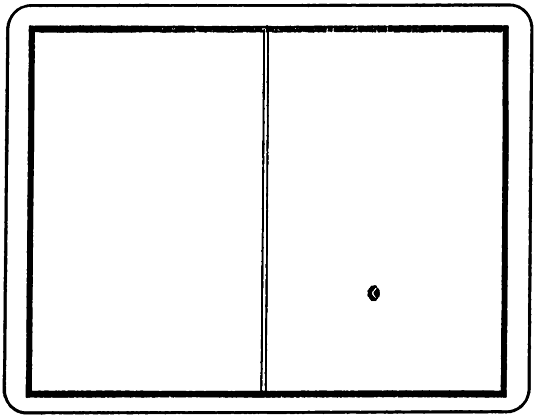


Fig. 8.6 : GET et PUT, le filet demeure intact.

(*) La fiche détaillée de l'instruction PUT figure à la page 334 du volume 2 de Pratique des IBM-PC et compatibles.

Bien que petit, ce programme nécessite toutefois quelques explications !... Avez-vous noté la façon d'incorporer la forme de la **BALLE** dans le tableau du même nom ?... On dessine la forme à l'emplacement où elle doit apparaître pour la première fois (ligne 70) et on l'incorpore à la ligne suivante dans le tableau. Du coup, la ligne 90 qui affiche le contenu du tableau dans la couleur complémentaire du fond va effacer la balle présente sur l'écran puisqu'une image complémentée par elle-même équivaut à un ensemble vide (vieux souvenirs des cours de maths modernes...).

Effaçant la balle à la ligne 90, on se retrouve à la dessiner à la ligne 110. C'est pourquoi vous allez trouver entre les deux (c'est-à-dire à la ligne 100) l'incrémentation des variables de position de la balle. En d'autres termes, la partie dynamique du programme qui va de la ligne 90 à la ligne 140 se trouve être inversée par rapport à ce qu'elle était dans l'ancien programme puisque, cette fois-ci, on commence par effacer la balle avant de la redessiner.

Remarque : *Sachant que l'image affichée par l'instruction PUT est référencée par rapport à son coin supérieur gauche, vous comprenez aisément pourquoi les formes graphiques le sont aussi !... Cela ne peut que simplifier la tâche du programmeur (c'est-à-dire la vôtre).*

> **Les instructions GET et PUT permettent d'animer des formes graphiques sans altérer le décor de fond de l'image.**

LE CONTRÔLE DES DÉPLACEMENTS À PARTIR DU CLAVIER.

L'exercice précédent n'ayant d'autre prétention que de vous expliciter le principe d'animation d'une forme graphique sur un décor de fond, nous allons cette fois-ci prendre les commandes en main (fini le pilotage automatique...) et contrôler la situation à partir du clavier.

Pour ce faire, nous allons choisir un nouvel exemple. Les idées ne manquant pas, que diriez-vous si l'on s'amusait à déplacer une main sur la surface de l'écran ?... A quoi cela peut-il servir ?... Mais par exemple à aller chercher des formes graphiques dans une bibliothèque de formes et à les mettre en place sur l'écran à l'endroit qui nous convient (comme pour dessiner un schéma d'électronique à partir des symboles des composants).

Quelles touches allons-nous utiliser pour déplacer la main ?... Il semble que les touches fléchées soient les plus pratiques d'emploi (pour les droitiers s'entend !...). Bien entendu, cette solution n'est pas la seule que l'on puisse imaginer. Les touches de fonctions peuvent également servir, tout comme les touches normales du clavier.
Voici donc un premier programme qui utilise les touches fléchées du clavier numérique pour contrôler les déplacements de la main :

```
10 DIM MAIN(9)
20 KEY OFF:SCREEN 1:CLS:COLOR 8,1:COULEUR=2
30 MAIN$="C=COULEUR;RFLFRBR5RGLBL3LGBR2RBR2RDL6GR7FBL2L6FR4BR2RGLBL2L2BFBR2R"
40 X=160:Y=100:PRESET(X,Y):DRAW MAIN$
50 GET(X,Y)-(X+9,Y+10),MAIN
60 PUT(X,Y),MAIN
70 IF ASCII=72 AND Y>0 THEN Y=Y-1
80 IF ASCII=75 AND X>0 THEN X=X-1
90 IF ASCII=77 AND X<310 THEN X=X+1
100 IF ASCII=80 AND Y<189 THEN Y=Y+1
110 PUT(X,Y),MAIN
120 ASCII=0:TOUCHE$=INKEY$:IF TOUCHE$="" THEN 60
130 IF ASC(TOUCHE$)=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1))
140 GOTO 60
```

Les déplacements de la main...

... dépendent de la touche enfoncée !

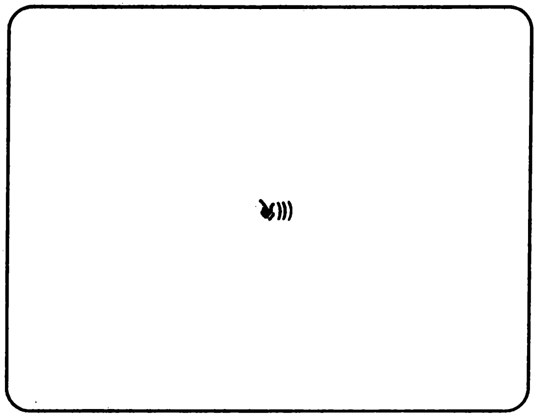


Fig. 8.7 : Main qui se déplace sous le contrôle du clavier

Que pensez-vous du résultat obtenu ?... Vous ne le trouvez pas génial !... J'avoue qu'il en est de même en ce qui me concerne, la main étant affectée d'un clignotement à cycle variable assez fatiguant à observer à la longue. De surcroît, les déplacements sont trop lents ; il faut pratiquement une trentaine de secondes pour aller d'un bord à l'autre de l'écran. Ce qu'il faudrait, c'est un système à vitesse variable, celle-ci dépendant du maintien de la touche en position enfoncée.

Nous allons donc résoudre les deux problèmes en les prenant un par un. Si l'on analyse les raisons qui font clignoter la **MAIN**, on constate qu'elles proviennent du fait qu'on exécute 4 lignes d'instructions (ce sont les lignes 70 à 100) entre le moment où celle-ci est effacée de l'écran (ligne 60) et celle où elle est redessinée (ligne 110). Qui plus est, ces 4 lignes comportent des tests en cascade qui ne font qu'en accroître le temps d'éxécution.

La solution ?... Elle est simple !... Elle consiste à mettre ces lignes ailleurs et à ne placer entre les lignes d'effacement et de dessin de la main que le strict minimum, à savoir le changement de valeur des variables X et Y. Le programme ci-dessous vous montre comment il faut procéder :

```
10 DIM MAIN(9)
20 KEY OFF:SCREEN 1:CLS:COLOR 8,1:COULEUR=1
30 MAIN$="C=COULEUR;RFLFRBR5RGLBL3LGBR2RBR2RDL6GR7FBL2L6FR4BR2RGLBL2L2BFBR2R"
40 X=160:Y=100:PRESET(X,Y):DRAW MAIN$
50 GET(X,Y)-(X+9,Y+10),MAIN
60 PUT(X,Y),MAIN
70 X=X+H:Y=Y+V
80 PUT(X,Y),MAIN
90 ASCII=0:TOUCHE$=INKEY$:IF TOUCHE$="" THEN 110
100 IF ASC(TOUCHE$)=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1))
110 H=0:V=0
120 IF ASCII=72 AND Y>0 THEN V=-1
130 IF ASCII=75 AND X>0 THEN H=-1
140 IF ASCII=77 AND X<310 THEN H=1
150 IF ASCII=80 AND Y<189 THEN V=1
160 GOTO 60
```

Cette ligne unique placée entre l'effacement et le dessin de la main...

... limite le clignotement de celle-ci à son strict minimum !

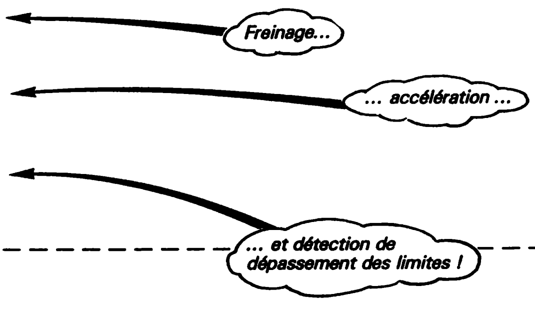
Tout d'abord, il vous faut rajouter les variables d'incrémentation H et V qui vont permettre de transférer les lignes de détermination des déplacements (lignes 120 à 150) loin de la ligne 70. Ce faisant vous ne conservez qu'une seule ligne d'instructions (la ligne 70) entre l'effacement et le dessin de la main, limitant ainsi le clignotement à son strict minimum.

Notez que cette nouvelle écriture nécessite la remise à zéro des incréments H et V (ligne 110) dès que vous cessez d'appuyer sur les touches fléchées, sans quoi la main se perd à l'infini (à moins que cela ne soit un effet recherché...).

Au tour maintenant du changement de vitesse des déplacements. Nous allons créer un effet d'accélération et de freinage progressif comme si la main était dotée d'une certaine inertie (c'est ce qu'on appelle avoir la main lourde...). Pour parvenir à un tel résultat, il suffit de conserver la structure du programme précédent en y remplaçant la ligne 110 de remise à zéro des incréments H et V par les lignes 110 à 140 de retour progressif à zéro (c'est en quelque sorte le freinage).

Côté accélération, celui-ci prend effet de la ligne 150 à la ligne 180, l'incrément H ou V subissant alors un accroissement de valeur aussi longtemps que le code ASCII demeure le même, c'est-à-dire aussi longtemps que la touche reste enfoncée :

```
10 DIM MAIN(9)
20 KEY OFF:SCREEN 1:CLS:COLOR 8,1:COULEUR=1
30 MAIN$="C=COULEUR;RFLFRBR5RGLBL3LGBR2RBR2RDL6GR7FBL2L6FR4BR2RGLBL2L2BFBR2R"
40 X=160:Y=100:PSET(X,Y):DRAW MAIN$
50 GET(X,Y)-(X+9,Y+10),MAIN
60 PUT(X,Y),MAIN
70 X=X+H:Y=Y+V
80 PUT(X,Y),MAIN
90 ASCII=0:TOUCHE$=INKEY$:IF TOUCHE$=""THEN 110
100 IF ASC(TOUCHE$)=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1)):GOTO 150
110 IF V<0 THEN V=V+1
120 IF H<0 THEN H=H+1
130 IF H>0 THEN H=H-1
140 IF V>0 THEN V=V-1
150 IF ASCII=72 THEN V=V-1
160 IF ASCII=75 THEN H=H-1
170 IF ASCII=77 THEN H=H+1
180 IF ASCII=80 THEN V=V+1
190 IF Y+V<0 THEN V=-Y
200 IF X+H<0 THEN H=-X
210 IF X+H>310 THEN H=310-X
220 IF Y+V>189 THEN V=189-Y
230 GOTO 60
```



Notez que cette nouvelle forme d'écriture oblige à dissocier les lignes de détection et de dépassement des limites (lignes 190 à 220) de celles d'augmentation de valeur des incréments. En effet, le freinage étant progressif, il s'en suit qu'il faut s'assurer à chaque parcours de la boucle d'animation si la main n'est pas susceptible de sortir des limites de l'écran à la suite de l'élan qui lui a été conféré.

Le contrôle des déplacements à partir du clavier nécessite l'emploi de variables d'incrémentation en complément des variables habituelles de positionnement de la forme graphique, de façon à minimiser le clignotement parasite dont cette dernière est affectée.

SÉQUENCE D'ANIMATION NÉCESSITANT PLUSIEURS FORMES GRAPHIQUES

Animer une forme graphique, c'est bien mais c'est peu !... Dans la réalité des choses, les logiciels graphiques emploient de nombreuses formes destinées (entre autres...) à créer l'illusion du mouvement d'un objet ou d'un personnage. Il faut donc aborder l'animation à un niveau plus élevé en considérant que celle-ci nécessite l'enchaînement de séquences dans lesquelles les formes graphiques se succèdent selon un ordre préétabli.

Pour illustrer ces propos, nous allons faire marcher un personnage de la gauche vers la droite de l'écran en simulant son mouvement de jambes et son balancement des bras. Bien que pouvant être décomposé en de multiples plans différents, nous nous limiterons de par la faible taille du personnage aux deux plans du mouvement que voici :

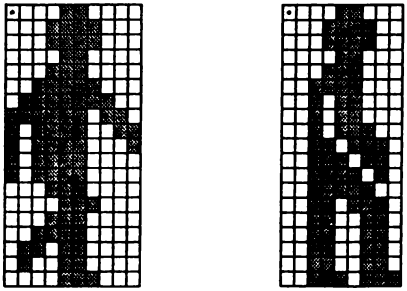


Fig. 8.8 : Décomposition du mouvement d'un personnage.

... que le programme suivant va nous dessiner et effacer à tour de rôle de façon à créer l'illusion du mouvement :

```
10 DIM PLAN1(15),PLAN2(15)
20 KEY OFF:SCREEN 1:COLOR 8,0:CLS:COULEUR=1
30 PLAN1$="C=COULEUR;BR4RFL3DR3GLGR2FL4GR6FLBL2L5DBR2R3BR3RDBL4L3BL2DBR2R3DL3BL2
   DOBFBR2R2FL2GR2DL3GRBR2RDLBL3DOBFBR2R2"
40 PLAN2$="C=COULEUR;BR4RFL3DR3GLGR2DL3DBR2RDLBL2DBR2R2FL2BL2LDR2BR2RDBL2L3DR4FL
   2BL2LDRBR3RDLBL3LDRBR3RDLBL3LDR2BR2R2"
50 LINE(0,119)-(319,119),2
60 X=0:Y=100:PRESET(X,Y):DRAW PLAN1$
70 GET(X,Y)-(X+9,Y+18),PLAN1
80 PUT(X,Y),PLAN1
90 X=X+2:PRESET(X,Y):DRAW PLAN2$
100 GET(X,Y)-(X+9,Y+18),PLAN2
110 PUT(X,Y),PLAN2
120 X=X+2:IF X>310 THEN END
130 PUT(X,Y),PLAN1
140 FOR T=1 TO 100:NEXT
150 PUT(X,Y),PLAN1
160 X=X+2:IF X>310 THEN END
170 PUT(X,Y),PLAN2
180 FOR T=1 TO 100:NEXT
190 GOTO 110
```

Effacement du plan n° 2... (ligne 110)

... suivi du dessin et de l'effacement du plan n° 1... (lignes 130 et 150)

... suivi à son tour du dessin du plan n° 2, etc, etc... (ligne 170)

Comme vous pouvez le constater de vous-même, il ne diffère guère des précédents. Après dessin du plan n° 1 (ligne 60) et son transfert dans le tableau **PLAN 1** (ligne 70), le plan n° 1 est effacé (ligne 80) et le plan n° 2 lui succède 2 pixels plus loin (ligne 90). Celui-ci est à son tour transféré dans le tableau **PLAN 2** (ligne 100) avant d'être effacé comme son prédécesseur (ligne 110).

Les deux plans étant chargés dans les tableaux **PLAN 1** et **PLAN 2**, ceux-ci se succèdent alors en alternance de façon à simuler la marche du personnage (lignes 120 à 190). Notez la présence de 2 ralentisseurs (lignes 140 et 180) destinés à empêcher celui-ci de partir dans une course éperdue...

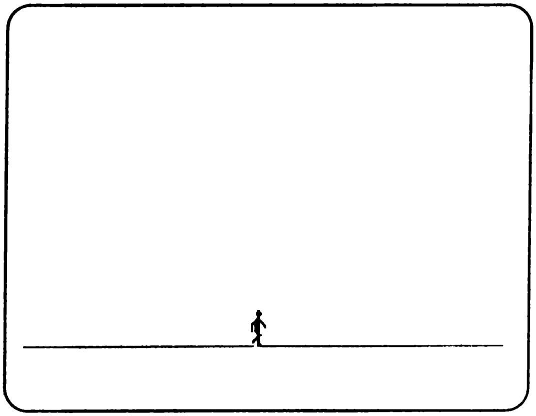


Fig. 8.9 : En voilà un qui ne se fatigue pas de marcher !

> **Pour créer une séquence d'animation, il suffit de placer chaque plan différent dans un tableau et de les afficher à tour de rôle en prenant à chaque fois la précaution d'effacer un tableau avant d'en dessiner un autre.**

CHANGEMENTS ALTERNÉS DE PLANS D'ANIMATION

Les deux exercices précédents vous ont montré, d'une part, comment contrôler au clavier les déplacements d'une forme unique et, d'autre part, comment animer les divers plans d'une forme (il y en avait deux). Dans le présent exercice, nous allons combiner les deux pour obtenir des déplacements (eux aussi contrôlés au clavier) d'une forme comportant plusieurs plans d'animation.

Comme vous pouvez le constater, la difficulté va croissante !... C'est pourquoi il vous est proposé pour illuster cette nouvelle étape de vous transformer en pilote (et même en constructeur...) d'hélicoptère. Cela vous tente ?... Alors, il n'y a pas à hésiter !... Vous relevez vos manches, faites jouer vos articulations des doigts et au travail !

Tout d'abord, vous allez commencer par dessiner les 2 plans du profil droit de l'appareil que voici :

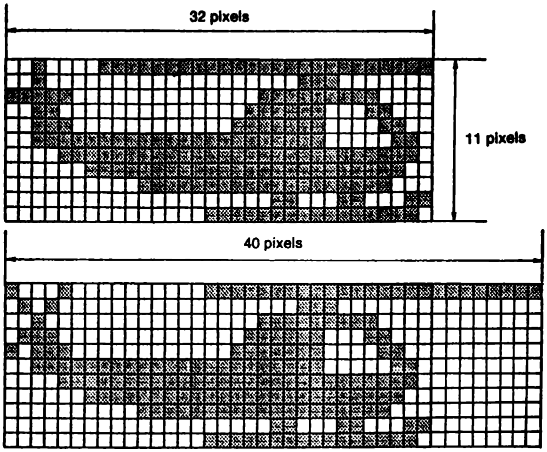


Fig. 8.10 : Plans 1 et 2 du profil droit de l'hélicoptère

Après le profil droit de l'hélicoptère, on pourrait être tenté de dessiner également les deux plans de la vue de face, ainsi que les deux plans du profil gauche. La place manquant, vous ne trouverez que les deux plans du profil droit dans le programme ci-dessous. Bien entendu, rien ne vous empêche d'y incorporer les plans manquants :

```
10 DIM HELICO1(22),HELICO2(28)
20 KEY OFF:SCREEN 1:COLOR 8,0:CLS:COULEUR=2
30 PLAN1$="C=COULEUR;BR2R0BR5R24BGBL6L2BL20L0BLBGR4BR15R8FLBL4L5BL15LDR2BR13R6BR
   5RFLBL6L21BFBRR26GL23BFBR3R18BR2BFLBL4LBL4LBL4BGR15"
40 PLAN2$="C=COULEUR;R0BR4R0BR11R24BGBL14L2BL19L0BL2FBR17R8FLBL4L5BL15L2GBR2R2BR
   13R6BR5RFLBL6L21BFBRR26GL23BFBR3R18BR2BFLBL4LBL4LBL4BGR15"
50 LINE(0,199)-(319,199),1
60 X=120:Y=188:PLAN=1
70 PRESET(X,Y):DRAW PLAN1$:GET(X,Y)-(X+31,Y+10),HELICO1:PUT(X,Y),HELICO1
80 PRESET(X,Y):DRAW PLAN2$:GET(X,Y)-(X+39,Y+10),HELICO2
90 IF PLAN=1 THEN PUT(X,Y),HELICO2:X=X+H:Y=Y+V:PUT(X,Y),HELICO1
100 IF PLAN=2 THEN PUT(X,Y),HELICO1:X=X+H:Y=Y+V:PUT(X,Y),HELICO2
110 PLAN=PLAN+1:IF PLAN>2 THEN PLAN=1
120 ASCII=0:TOUCHE$=INKEY$:IF TOUCHE$=""THEN 140
130 ASCII=ASC(TOUCHE$):IF ASCII=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1))
140 IF ASCII=0 THEN H=0:V=0
150 IF ASCII=71 THEN H=-2:V=-2
160 IF ASCII=72 THEN H=0:V=-2
170 IF ASCII=73 THEN H=2:V=-2
180 IF ASCII=75 THEN H=-2:V=0
190 IF ASCII=77 THEN H=2:V=0
200 IF ASCII=79 THEN H=-2:V=2
210 IF ASCII=80 THEN H=0:V=2
220 IF ASCII=81 THEN H=2:V=2
230 IF Y=0 AND V=-2 THEN V=0
240 IF X=0 AND H=-2 THEN H=0
250 IF X=280 AND H=2 THEN H=0
260 IF Y=188 AND V=2 THEN V=0
270 GOTO 90
```

Avant d'analyser ce programme, essayez-le !... Dès son exécution, vous obtenez un hélicoptère moteurs tournants, n'attendant plus que le bon vouloir de son pilote pour décoller. Où se trouve le manche à balai ?... Ne cherchez plus, ce sont les 8 touches du clavier numérique qui font office !... Appuyez donc sur la touche fléchée montante et regardez-vous décoller !... C'est-y pas génial ?

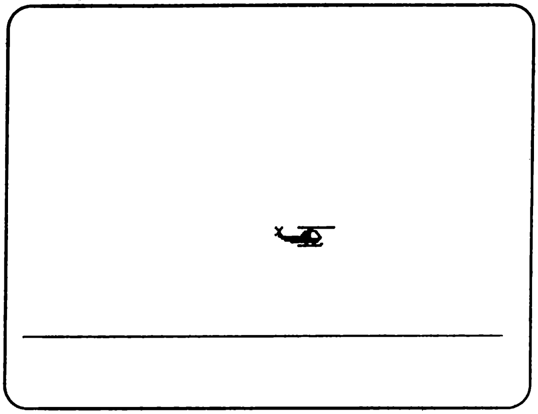


Fig. 8.11 : Apprenez à piloter un hélicoptère pour pas cher !

Pour revenir à des choses sérieuses, le programme que vous venez de taper ne diffère de ces prédécesseurs que par les lignes 90 à 110. Or, si vous regardez de près ces trois lignes, ainsi que la fin de la ligne 60, vous allez constater qu'à chaque parcours de la boucle vous dessinez un PLAN différent du profil droit de l'hélicoptère (lignes 90 et 100), le changement de plan se faisant par incrémentation de la variable PLAN à la ligne 110. Tout le secret est là !...

De surcroît, cette technique de changement de plan par incrémentation n'est nullement limitée à deux plans. Vous pouvez selon votre application envisager autant de plans différents que la mémoire de votre IBM-PC vous y autorise.

Dans une animation, le changement de plans d'un objet ou d'un personnage doit s'effectuer de préférence à chaque parcours de la boucle.

ENTRÉE ET SORTIE PROGRESSIVE DE L'ÉCRAN

Vous n'avez pas été sans remarquer l'impossibilité pour les instructions PUT et DRAW de dessiner partiellement (ou totalement) une forme en-dehors de l'écran, excluant ainsi toute possibilité d'entrée ou de sortie progressive de celle-ci sur l'écran. Or, quand on examine de près les logiciels graphiques du commerce, on constate qu'une telle possibilité existe !... C'est donc le preuve qu'il existe des méthodes susceptibles de produire un tel effet.

Ces méthodes font pour la plupart d'entre-elles appel au langage machine, seul langage capable d'exécuter suffisamment rapidement ce type d'opération. Comme il ne peut être question dans cet ouvrage d'aborder d'autre langage que le Basic, il ne nous reste plus qu'à nous rabattre sur la méthode archaïque de découpage en tranches des formes graphiques.

Pour illustrer cette méthode de découpage en tranches, nous avons choisi de faire passer devant l'écran d'abord une voiture, puis plusieurs et pour finir, un véritable autoroute un jour de grand départ en vacances. Bien sûr, il faut disposer pour cela d'un modèle de série susceptible d'être reproduit en un grand nombre d'exemplaires. C'est pourquoi notre choix s'est porté sur la voiture de l'année que voici :

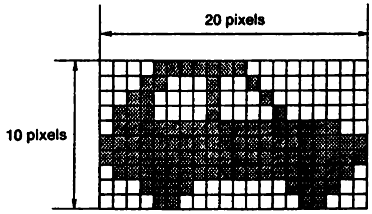


Fig. 8.12 : Voiture de l'année remarquée au dernier salon de l'auto

Cette voiture faisant 20 pixels de longueur hors-tout, nous avons choisi de la découper en tranches d'épaisseur croissante de 5 pixels de pas (5, 10 et 15 pixels) en nous alignant sur le pare-choc avant (entrée progressive) ou sur le pare-choc arrière (sortie progressive). Cela nous donne les découpes suivantes :

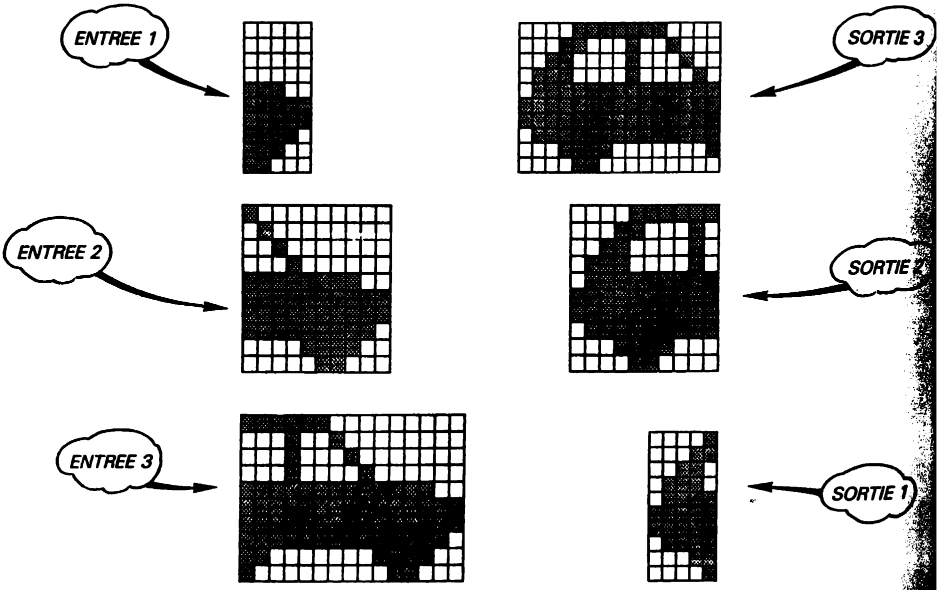


Fig. 8.13 : Découpage en tranches de la voiture

Ceci étant, il ne nous reste plus qu'à écrire le programme capable d'effectuer le découpage que nous venons de définir et d'enchaîner ces différentes découpes de façon à donner l'illusion d'une entrée et d'une sortie progressive de la voiture sur l'écran :

```
10 DIM VOITURE(13)
20 DIM ENTREE1(5),ENTREE2(8),ENTREE3(10)
30 DIM SORTIE1(5),SORTIE2(8),SORTIE3(10)
40 VOITURE$="C=COULEUR;BR4R6FLOBL3LOBL4LGRBR5ROBR4FBL5LOBL5L2DR16BRBFL19DR19GL17
   BFBRR3BR8R3GLBL10L"
50 KEY OFF:SCREEN 1:COLOR 8,0:CLS:COULEUR=3
60 X=0:Y=100:PRESET(X,Y):DRAW VOITURE$
70 GET(X+15,Y)-(X+19,Y+9),ENTREE1
80 GET(X+10,Y)-(X+19,Y+9),ENTREE2
90 GET(X+5,Y)-(X+19,Y+9),ENTREE3
100 GET(X,Y)-(X+19,Y+9),VOITURE
110 GET(X,Y)-(X+14,Y+9),SORTIE3
120 GET(X,Y)-(X+9,Y+9),SORTIE2
130 GET(X,Y)-(X+4,Y+9),SORTIE1
140 IF X<305 THEN PUT(X,Y),VOITURE ELSE GOSUB 200
150 X=X+5:IF X=320 THEN X=0
160 IF X<305 THEN PUT(X,Y),VOITURE ELSE GOSUB 200
170 FOR T=1 TO 100:NEXT
180 GOTO 140
190 'Sous-programme d'entrée et de sortie progressive-------------------------
200 IF X=305 THEN PUT(0,Y),ENTREE1:PUT(X,Y),SORTIE3
210 IF X=310 THEN PUT(0,Y),ENTREE2:PUT(X,Y),SORTIE2
220 IF X=315 THEN PUT(0,Y),ENTREE3:PUT(X,Y),SORTIE1
230 RETURN
```

Découpe en tranches de la voiture ... (lignes 70 à 90 et 110 à 130)

... et leur affichage en bordure d'écran ! (lignes 200 à 220)

Le moins que l'on puisse dire en examinant ce programme, c'est que la découpe ne passe pas inaperçue !... Pas moins de 6 lignes y sont consacrées, les lignes 70 à 90 assurant les découpes en correspondance avec le bord gauche de l'écran (tableaux **ENTREE 1** à **ENTREE 3** et les lignes 110 à 130 les découpes en correspondance avec le bord droit de l'écran (tableaux **SORTIE 1** à **SORTIE 3**).

Remarquez aux lignes 140 et 160 la méthode employée pour dessiner ces découpes !... Tant que X reste inférieur à 305, les lignes 140 et 160 effacent et dessinent la voiture dans sa totalité. Ce n'est que lorsque X devient égal ou supérieur à 305 que l'on se dirige vers le sous-programme 200 chargé de dessiner (ou d'effacer selon le cas) les découpes situées en bordure d'écran (lignes 200 à 220).

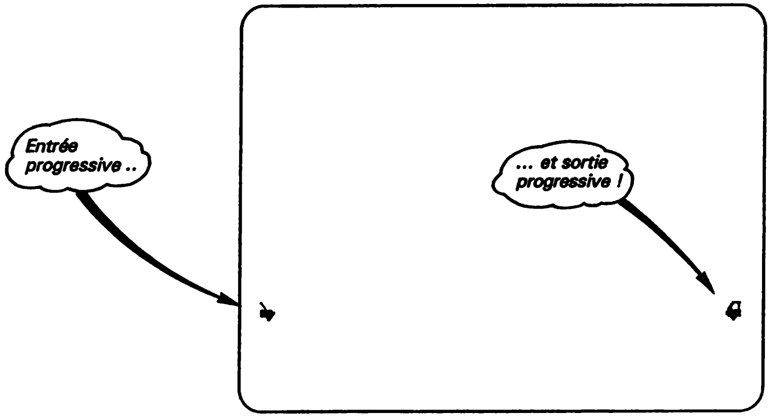


Fig. 8.14 : Entrée et sortie progressive d'une voiture de l'écran.

Comme vous pouvez le constater par vous-même, cette méthode de « saucissonnage » des formes graphiques n'est utilisable qu'à la condition de tricher. Pas question par exemple de découper la forme en tranches d'un pixel d'épaisseur !... Le nombre de tranches serait trop important et la longueur du programme aussi. C'est pourquoi le pas de progression (et donc de découpe) a été agrandi à 5 pixels.

Ceci dit, nous allons conclure ce chapitre sur l'animation graphique (on pourrait écrire un livre entier sur le sujet) en vous présentant une portion d'autoroute un jour où « Bison Futé » conseille fortement d'emprunter les routes secondaires :

```
10 DIM VOITURE(13),VOITURES(188)
20 DIM ENTREE1(5),ENTREE2(8),ENTREE3(10)
30 DIM SORTIE1(5),SORTIE2(8),SORTIE3(10)
40 VOITURE$="C=COULEUR;BR4R6FL0BL3L0BL4LGRBR5R0BR4FBL5L0BL5L2DR16BRBFL19DR19GL17
   BFBRR3BR8R3GLBL10L"
50 KEY OFF:SCREEN 1:COLOR 8,0:CLS:COULEUR=2
60 FOR Y=100 TO 130 STEP 15:FOR X=0 TO 300 STEP 25:
   PRESET(X,Y):DRAW VOITURE$:NEXT X,Y
70 X=0:Y=100:GET(X,Y)-(X+299,Y+9),VOITURES
80 GET(X+15,Y)-(X+19,Y+9),ENTREE1
90 GET(X+10,Y)-(X+19,Y+9),ENTREE2
100 GET(X+5,Y)-(X+19,Y+9),ENTREE3
110 GET(X,Y)-(X+19,Y+9),VOITURE
120 GET(X,Y)-(X+14,Y+9),SORTIE3
130 GET(X,Y)-(X+9,Y+9),SORTIE2
140 GET(X,Y)-(X+4,Y+9),SORTIE1
150 X=X1:Y=100:PUT(X,Y),VOITURES:GOSUB 270
160 X1=X1+15:IF X1>=25 THEN X1=X1-25
170 X=X1:PUT(X,Y),VOITURES:GOSUB 270
180 X=X2:Y=115:PUT(X,Y),VOITURES:GOSUB 270
190 X2=X2+10:IF X2>=25 THEN X2=X2-25
200 X=X2:PUT(X,Y),VOITURES:GOSUB 270
210 X=X3:Y=130:PUT(X,Y),VOITURES:GOSUB 270
220 X3=X3+5:IF X3>=25 THEN X3=X3-25
230 X=X3:PUT(X,Y),VOITURES:GOSUB 270
240 FOR T=1 TO 100:NEXT
250 GOTO 150
260 'Sous-programme d'entrée et de sortie progressive------------------------
270 IF X=0 THEN PUT(300,Y),VOITURE
280 IF X=5 THEN PUT(305,Y),SORTIE3
290 IF X=10 THEN PUT(0,Y),ENTREE1:PUT(310,Y),SORTIE2
300 IF X=15 THEN PUT(0,Y),ENTREE2:PUT(315,Y),SORTIE1
310 IF X=20 THEN PUT(0,Y),ENTREE3
320 RETURN
```

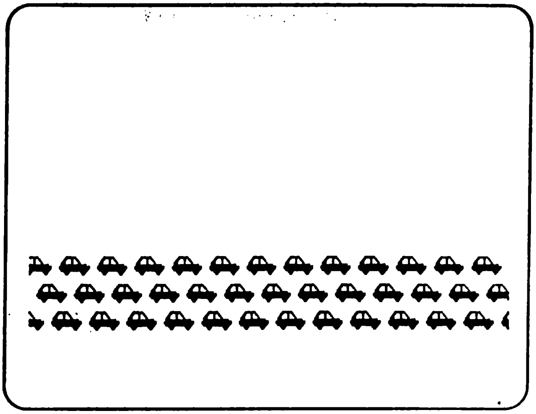


Fig. 8.15 : Autoroute un dimanche soir d'été...

Au vu des résultats obtenus, que peut-on en conclure ?... Que nous venons d'atteindre avec cet exemple les limites du Basic en matière de vitesse d'exécution. Pour passer au-dessus de ce seuil, il vous faut compiler vos programmes (on en parlera dans un prochain chapitre) et, si cela ne suffit pas, il y a lieu de changer carrément de langage et de se diriger vers le Turbo Pascal(*) ou le langage machine du 8088(*).

L'entrée et la sortie progressive de l'écran d'une forme graphique est difficilement réalisable sous Basic. Le Turbo Pascal ou le langage machine sont mieux adapté à ce type d'animation.

(*) Lire à ce sujet les ouvrages :
8088 assembleur de H. Lilen
Pratique du Turbo Pascal de J.J. Meyer.

Exercices sur le chapitre 8

1. Pourquoi est-il préférable de débuter une séquence d'animation par l'effacement de la première forme ?

   ☐ Parce que cela réduit le clignotement inhérent au temps d'absence de la forme sur l'écran.
   ☐ Parce que c'est la seule disposition possible lorsque l'animation porte sur plusieurs formes.

2. L'instruction DRAW permet-elle de conserver le décor de fond sur lequel se déplace une forme graphique ?

   ☐ Oui, car elle dessine la forme dans la couleur complémentaire du fond.
   ☐ Non, car elle est incapable de dessiner dans la couleur complémentaire du fond.

3. Doit-on dimensionner un tableau dans lequel on veut mémoriser une portion d'image ?

   ☐ Oui, dans tous les cas !
   ☐ Non, si la taille de la forme ne nécessite pas un tableau de plus de 10 indices.

4. Que se passe-t-il lorsqu'une portion d'image dessinée par l'instruction PUT n'est pas contenue dans l'écran ?

   ☐ L'ordinateur émet un message d'erreur
   ☐ L'ordinateur ignore purement et simplement les pixels situés en dehors de l'écran.
   ☐ L'ordinateur affiche sur le bord opposé les pixels situés en dehors de l'écran

5. Pourquoi faut-il aller lire le second octet délivré par l'instruction INKEY$?

   ☐ Parce qu'il correspond au code ASCII supérieur de la touche enfoncée.
   ☐ Parce qu'il correspond au code des touches du clavier numérique.

6. Quelle est l'instruction Basic la mieux adaptée à produire une entrée ou une sortie progressive d'une forme de l'écran ?

   ☐ Il n'y en a pas !
   ☐ L'instruction PUT qui permet de dessiner une image à cheval sur un bord d'écran.

Réponses page 173

Solution des exercices sur le chapitre 7

1. 8 × 8
2. à écrire du texte au format 80 colonnes en moyenne résolution ET à écrire du texte au format 160 colonnes en haute résolution ET à écrire du texte à lecture verticale en haute résolution.
3. En chargeant la police de caractères graphiques dénommée GRAFTABL.
4. Elle indique la couleur d'un pixel, qu'il soit allumé ou non.
5. Bien que l'on puisse employer une police de caractères élargis, le plus simple consiste à doubler la largeur de chaque pixel.
6. La seule solution consiste à employer une police de caractères étroits.

CHAPITRE IX

TRAITEMENT DES IMAGES

Après nous être intéressés aux formes graphiques qui n'occupaient, somme toute, qu'une faible portion d'écran, nous allons dans le présent chapitre aborder les images remplisant l'écran dans sa totalité en passant en revue quelques-uns des traitements qu'il est possible de leur faire subir. C'est ainsi qu'après vous avoir montré comment redéfinir les coordonnées graphiques de l'écran, il vous sera expliqué comment obtenir un effet de zoom électronique sur vos images ou encore comment réaliser l'effet miroir, etc, etc..., sans oublier l'essentiel, à savoir la sauvegarde, le rappel et l'impression sur papier de vos chef-d'œuvres graphiques.

Menu du chapitre 9

Redéfinition des limites d'écran.
L'effet de zoom que procure l'instruction WINDOW.
Une véritable loupe électronique
Découpage et collage
L'effet miroir
Chargement et sauvegarde d'une image
Recopie des écrans graphiques

Quel est votre choix : ■

REDÉFINITION DES LIMITES D'ÉCRAN

WINDOW

Vous pensiez immuables les coordonnées graphiques de l'écran et en fait il n'en est rien !... Le Basic Microsoft de votre IBM-PC (ou de ses compatibles) vous offre la possibilité, au travers de l'instruction WINDOW (*), de modifier à votre guise les co-ordonnées graphiques de l'écran et ce, dans la limite de 10^{-38} à 10^{+38}, ce qui est tout simplement fantastique !

(*) La fiche détaillée de l'instruction WINDOW figure à la page 343 du volume 2 de Pratique des IBM-PC

Ainsi, au lieu du traditionnel écran graphique à moyenne résolution de 320 × 200 pixels, un simple :

WINDOW agrandit ou rétrécit l'écran !

```
10 KEY OFF:SCREEN 1:COLOR 8,0:CLS
20 WINDOW (0,0)-(999,999)
```

... vous transforme l'écran en un carré de 1000 × 1000 unités, la preuve en est la ligne que voici :

```
30 LINE(0,0)-(319,199),1,B
```

... qui, au lieu de dessiner un cadre tout autour de l'écran (comme à l'accoutumée...), limite son action à l'angle inférieur gauche de l'écran. Or, ceci est tout ce qu'il y a de plus normal puisque le rectangle que l'on cherche à obtenir ne représente plus désormais que le 1/3 de l'écran en longueur (approximativement) et le 1/5 en hauteur. C'est dire sa petitesse !

Pour obtenir de nouveau un cadre tout autour de l'écran, il vous faut prendre en compte les nouvelles valeurs limites instaurées par l'instruction WINDOW et écrire :

```
30 LINE(0,0)-(999,999),1,B
```

Est-ce à dire que l'instruction WINDOW permet d'augmenter le nombre de pixels de l'écran ?... Non, bien entendu !... Cela veut tout simplement dire que les limites de l'écran qui étaient confondues avec le nombre de pixels de ce dernier ne le sont plus désormais et que, de 320 × 200, celles-ci ont été portées à 1000 × 1000 unités dans le précédent exemple.

Tout ceci est bien beau, pensez-vous, mais à quoi cela peut-il servir ?... Mais par exemple à dessiner une fonction par rapport à un repère orthogonal classique !... Ainsi, sans instruction WINDOW, il vous faudrait écrire pour afficher la courbe représentative du trinome du second degré $Y = X^2 - 3X + 2$ dans l'intervalle [-1, +4], le point de coordonnées 0, 0 étant placé au centre de l'écran :

```
10 KEY OFF:SCREEN 1:COLOR 8,0:CLS
20 X0=160:Y0=100:ECHELLE=20
30 LINE(0,Y0)-(319,Y0),2
40 LINE(X0,0)-(X0,199),2
50 FOR I=-7 TO 7:LINE(X0+I*ECHELLE,Y0-2)-(X0+I*ECHELLE,Y0+2),2:NEXT
60 FOR I=-5 TO 5:LINE(X0-2,Y0+I*ECHELLE*5/6)-(X0+2,Y0+I*ECHELLE*5/6),2:NEXT
70 FOR X=-1 TO 4 STEP 1/100
80 Y=X^2-3*X+2
90 PSET(X0+X*ECHELLE,Y0-Y*ECHELLE*5/6),1
100 NEXT
```

Dessin de la courbe représentative de la fonction $Y = X^2 - 3X + 2$ sans...

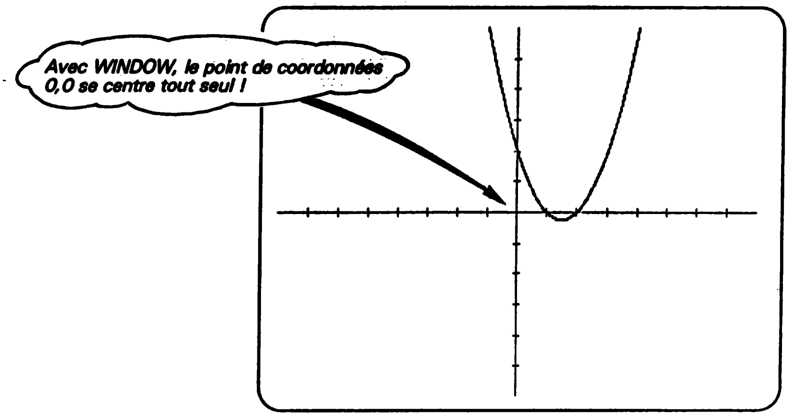


Fig. 9.1 : Courbe représentative de la fonction Y = $X^2 - 3X + 2$

Lourd, n'est-ce pas ?... Tandis qu'avec l'instruction WINDOW, l'écriture se simplifie quasiment de moitié en ceci :

```
10 KEY OFF:SCREEN 1:COLOR 8,0:CLS
20 WINDOW(-8,-6)-(8,6)
30 LINE(-8,0)-(8,0),2
40 LINE(0,-6)-(0,6),2
50 FOR I=-7 TO 7:LINE(I,-.1)-(I,.1),2:NEXT
60 FOR I=-5 TO 5:LINE(-.1,I)-(.1,I),2:NEXT
70 FOR X=-1 TO 4 STEP 1/100
80 Y=X^2-3*X+2
90 PSET(X,Y),1
100 NEXT
```

En regardant de plus près ce listage, vous allez constater la disparition des variables X0 et Y0 destinées à centrer les axes orthogonaux sur l'écran, de même que la variable d'ECHELLE qui désormais fait partie intégrante des paramètres de l'instruction WINDOW. En effet, une largeur d'écran de 16 unités (−8 à +8) multipliée par une ECHELLE de 20 équivaut bien à 320 unités, tandis qu'une hauteur de 12 (−6 à +6) multipliée par la même ECHELLE équivaut à 240 unités qui, à leur tour multipliées par le facteur de correction de 5/6 redonne bien les 200 unités usuelles.

Maintenant, regardez bien les paramètres de l'instruction WINDOW. Vous ne remarquez rien d'anormal ?... Pourtant, cela devrait vous sautez aux yeux !... Les paramètres de cette instruction qui fixent les coordonnées des deux angles opposés de l'écran, le font en débutant par une valeur négative pour Y (−6) alors que manifestement les valeurs positives de Y se situent dans le haut de l'écran. En fait, cela tient à ce que l'instruction WINDOW instaure une orientation cartésienne des axes, que l'on ait choisi de débuter par le coin supérieur gauche ou non.

SCREEN

Notez que cette inversion est bien pratique car elle permet de graduer l'axe des Y comme en mathématiques, le sens croissant des valeurs étant désormais dirigé vers le haut de l'écran. Si toutefois une telle disposition ne vous convenait pas, vous pourriez toujours rétablir celle d'origine en incorporant un **SCREEN** entre l'instruction **WINDOW** et les paramètres. Ainsi, l'incorporation d'un **SCREEN** dans la ligne 20 du programme précédent...

```
20 WINDOW SCREEN(-8,-6)-(8,6)
```

... a pour conséquence d'inverser l'orientation de l'axe des Y et donc de dessiner la fonction la tête en bas !

> **L'instruction WINDOW permet de redéfinir les coordonnées limites des écrans graphiques à moyenne et à haute résolution.**

L'EFFET DE ZOOM QUE PROCURE L'INSTRUCTION WINDOW

Une deuxième application fort intéressante de l'instruction **WINDOW** est l'effet d'agrandissement ou de réduction qu'elle apporte sur des dessins réalisés à l'aide d'instructions telles que **LINE** et **CIRCLE**.

Pour illustrer cet effet d'agrandissement (la réduction présentant moins d'intérêt...), voici un petit exemple de cartographie aérienne d'une ville américaine (devinez laquelle...) :

```
10 KEY OFF:SCREEN 1:COLOR 8,0:CLS
20 ZOOM=1
30 WINDOW SCREEN(0,0)-(320/ZOOM,240/ZOOM)
40 FOR Y=1 TO 201 STEP 40:FOR X=1 TO 281 STEP 40
50 LINE(X,Y)-(X+36,Y+36),2,B:NEXT X,Y
```

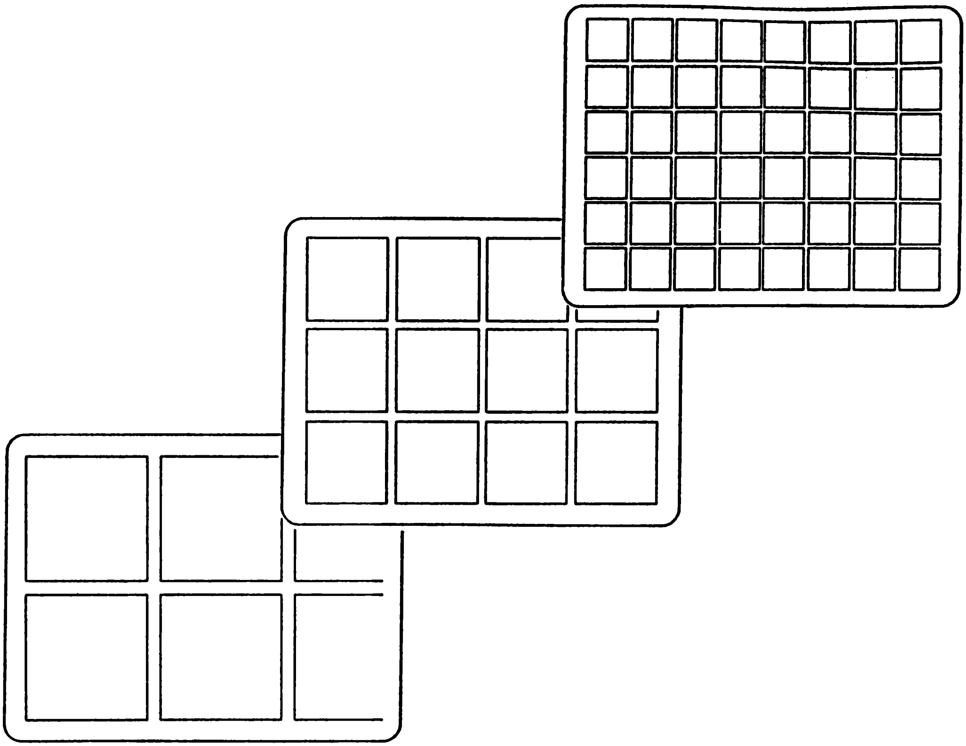


Fig. 9.2 : Exemple d'agrandissement obtenu avec l'instruction WINDOW.

Intéressant, comme effet d'agrandissement, n'est-ce pas ?... Or ce dernier est obtenu en modifiant la valeur de la variable **ZOOM** (ligne 20) qui, passant de 1 à 3, réduit à son tour les limites de l'écran graphique dans la même proportion et ce, par l'intermédiaire de l'instruction **WINDOW** de la figure 30.

De ce fait, il devient de plus en plus difficile de loger sur l'écran la totalité des 48 carrés de 40 unités de côté que dessinent les lignes 40 et 50, ce qui détermine l'effet d'agrandissement recherché. Pour ne prendre qu'un exemple, la valeur ZOOM = 2 réduit les limites d'écran à 320/2 = 160 et 240/2 = 120, ce qui ne permet plus de loger que 4 × 3 = 12 carrés sur l'écran au lieu des 48 du début. Ceci a donc pour effet d'agrandir par 2 les dimensions de chacun des carrés, les faisant ainsi apparaître plus grands.

Petite remarque en passant !... Avez-vous noté la valeur inhabituelle de 240 employée dans l'instruction **WINDOW** pour définir la hauteur de l'écran ?... Cela n'est rien d'autre que la valeur usuelle de 200 pixels à laquelle a été incorporé le facteur de correction de 5/6 (sinon les carrés apparaîtraient plus hauts que larges...).

L'instruction WINDOW procure un effet de zoom sur les figures dessinées avec les instructions LINE et CIRCLE.

UNE VÉRITABLE LOUPE ÉLECTRONIQUE

Bien que fort intéressant, l'effet de zoom que procure l'instruction **WINDOW** est très limité en application, puisqu'il est totalement inopérant sur les formes graphiques et les images. Vous ne me croyez pas ?... Essayez-donc d'agrandir le **CARRÉ** de 8 pixels de côté avec les quelques lignes que voici :

A votre avis, ce carré de 8 pixels de côté va-t-il remplir tout l'écran ?

```
10 KEY OFF:SCREEN 1:COLOR 8,0:CLS
20 CARRE$="C2R7DL7DR7DL7DR7DL7DR7DL7"
30 WINDOW(0,0)-(7,7)
40 PRESET(0,0):DRAW CARRE$
```

Ce n'est pas brillant, n'est-ce pas ?... Non seulement le carré ne remplit pas tout l'écran, mais de surcroît on n'en voit plus que la ligne du haut !... Et ne croyez pas améliorer la situation avec la lettre code S du sous-langage **DRAW** !... L'effet n'en serait que plus désastreux.

Et celui-ci ?

```
10 KEY OFF:SCREEN 1:COLOR 8,0:CLS
20 CARRE$="S100C2R7DL7DR7DL7DR7DL7DR7DL7"
30 PRESET(0,0):DRAW CARRE$
```

Il faut vous rendre à l'évidence, il n'existe pas d'instruction spécifique pour agrandir une forme graphique et, a fortiori, une image !

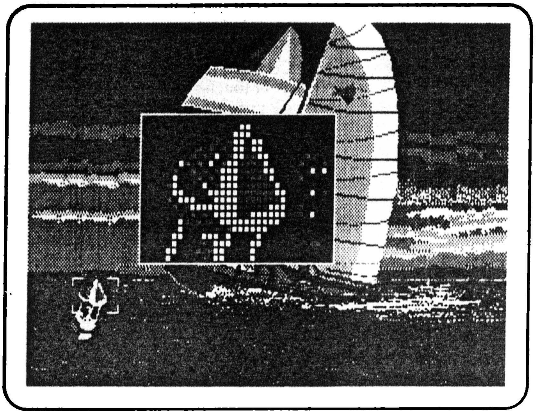
Qu'à cela ne tienne, nous allons la fabriquer ensemble !... Imaginez que vous disposiez d'une instruction (ou plutôt d'un sous-programme...) capable d'agrandir une portion d'image, comme dans les deux exemples ci-contre.

Impressionnant, n'est-ce pas ?... Or, ceci est à votre portée moyennant quelques lignes de programme, une image de fond et un peu d'huile de coude.
Tout d'abord, que faut-il ?... Un viseur, genre appareil photo, que l'on déplace sur l'écran à l'aide des touches fléchées. Aussitôt dit, aussitôt fait, le voici :

Dessin du viseur d'agrandissement... (lignes 30 à 50)

... qu'on déplace avec les touches fléchées ! (lignes 120 à 150)

```
10 DIM VISEUR(40),FOND(40),IMAGE(685)
20 KEY OFF:SCREEN 1:COLOR 8,0:CLS
30 LINE(X,Y)-(X+31,Y+19),,B
40 LINE(X+8,Y)-(X+23,Y+19),0,B
50 LINE(X,Y+5)-(X+31,Y+14),0,B
60 GET(X,Y)-(X+31,Y+19),VISEUR:CLS
70 BLOAD"REGATE.IMA"
80 GET(X,Y)-(X+31,Y+19),FOND:PUT(X,Y),VISEUR,OR
90 ASCII=0:TOUCHE$=INKEY$:IF TOUCHE$="" THEN 90
100 ASCII=ASC(TOUCHE$):IF ASCII=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1))
110 PUT(X,Y),FOND,PSET
120 IF ASCII=72 AND Y>0 THEN Y=Y-1
130 IF ASCII=75 AND X>0 THEN X=X-1
140 IF ASCII=77 AND X<288 THEN X=X+1
150 IF ASCII=80 AND Y<180 THEN Y=Y+1
160 IF ASCII<>32 THEN 80
```



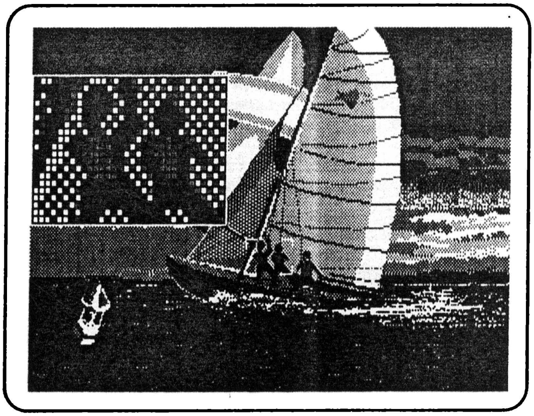


Fig. 9.3 : Voici ce qu'il est possible d'obtenir avec une loupe électronique

Comme vous pouvez le constater, ce viseur (ou plutôt le programme qui permet de l'obtenir...) nécessite quelques explications détaillées. Tout d'abord la ligne 10 dimensionne le tableau du **VISEUR** et celui de l'image de **FOND** sur laquelle il passe. Pourquoi deux tableaux pour cela, vous demandez-vous, alors qu'il suffit de dessiner le viseur dans la couleur complémentaire du fond pour que ce dernier ne soit pas altéré ?

Que répondre à cette remarque pertinente, si ce n'est que le résultat n'est pas beau à voir (mais alors, pas du tout...) car les bords du **VISEUR** que dessinent les lignes 30 à 50 apparaissent alors comme une succession de points de différentes couleurs.

PUT option OR

En fait, ce que l'on souhaite, c'est un **VISEUR** d'une seule et unique couleur !... Dans ces conditions, pas question d'employer l'astuce du dessin dans la couleur complémentaire. Ce qu'il faut, c'est dessiner le viseur par dessus l'image, ce que permet l'option **OR** de l'instruction **PUT** (*), non sans avoir au préalable sauvegardé avec un **GET** la portion d'image située sous le viseur (ligne 80).

Ceci fait, le reste est très classique !... Les lignes 90 et 100 assurent la reconnaissance de la touche frappée au clavier, le programme tournant en boucle sur la ligne 90 tant qu'aucune touche n'est enfoncée. Puis, c'est la ligne 110 qui efface le viseur en redessinant le **FOND** d'image et les lignes 120 à 150 assurent le déplacement du viseur. Quant à la ligne 160, elle boucle sur la ligne 80 tant que personne ne se hasarde à appuyer sur la barre d'espacement.

Pourquoi la barre d'espacement ?... Parce que c'est elle qui donne accès au programme de loupe électronique qu'il nous reste encore à écrire (mais c'est le plus intéressant...) :

C'est uniquement cette ligne qui réalise la loupe électronique !

```
170 IF X>=144 AND Y<90 THEN XL=X-144:YL=Y+27
180 IF X<144 AND Y<90 THEN XL=X+46:YL=Y+27
190 IF X<144 AND Y>=90 THEN XL=X+46:YL=Y-90
200 IF X>=144 AND Y>=90 THEN XL=X-144:YL=Y-90
210 GET(XL,YL)-(XL+130,YL+82),IMAGE
220 LINE(XL,YL)-(XL+130,YL+82),,B:LINE(XL+1,YL+1)-(XL+129,YL+81),0,BF
230 FOR J=0 TO 19:FOR I=0 TO 31:COULEUR=POINT(X+I,Y+J):
    LINE(XL+2+I*4,YL+2+J*4)-(XL+4+I*4,YL+4+J*4),COULEUR,BF:NEXT I,J
240 PUT(X,Y),VISEUR,OR
250 TOUCHE$=INKEY$:IF TOUCHE$="" THEN 250
260 PUT(XL,YL),IMAGE,PSET:GOTO 90
```

Je vous entends déjà vous exclamer : « C'est tout ? ». Eh oui, c'est tout !... Ces quelques lignes suffisent à réaliser une loupe électronique de rapport d'agrandissement égal à 4. En fait, pour tout vous avouer, c'est la ligne 230 qui en est le cœur !... Remarquez comment s'effectue l'analyse de la **COULEUR** de chacun des 32 × 20 pixels du viseur à l'aide de l'instruction **POINT**. Remarquez également comment chaque point d'image est redessiné sous la forme d'un carré plein de même **COULEUR**.

Le reste ?... Booaff !... Il s'agit pour les lignes 170 à 200 d'éviter de dessiner la loupe par-dessus le viseur (ce sont des choses qui ne se font pas...). La ligne 210, quant à elle, sauvegarde la portion d'**IMAGE** que la loupe va écraser, histoire de pouvoir la rétablir sans altération dès que l'on appuie sur une touche quelconque (lignes 250 et 260). La ligne 210 dessine en jaune sur fond noir le cadre de la loupe et pour finir, la ligne 240 redessine le **VISEUR** quand l'analyse des pixels est terminée (sinon, c'est le pourtour du viseur qui apparaît agrandi...).

L'instruction WINDOW est incapable d'agrandir ou de réduire la taille d'une forme graphique ou d'une image. Il faut pour cela écrire un programme spécifique.

(*) La fiche détaillée de l'instruction PUT est décrite à la page 334 du volume 2 de Pratique des IBM-PC.

DÉCOUPAGE ET COLLAGE

Puisque l'on est dans les programmes spécifiques, cela vous dirait-il d'apprendre à faire du découpage et du collage d'images... sans ciseaux ni colle, bien entendu ?... Comme vous pouvez vous en douter, il n'y a pas dans le Basic Microsoft d'instructions spécifiques telles que CUT et PASTE pour réaliser cela. Mais, est-ce vraiment gênant puisque, hormis le temps d'éxécution qui va forcément être plus long, quelques lignes d'instructions vont nous permettre d'aboutir au même résultat et ce, grâce aux instructions GET et PUT ?

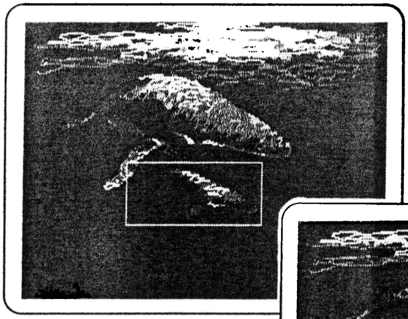
Si votre réponse à la question précédente est non, c'est-à-dire si vous considérez qu'un bon vieux programme Basic fait aussi bien l'affaire qu'un jeu d'instructions manquantes, voici comment il vous faut procéder. Tout d'abord, il vous importe de savoir qu'il y a 4 phases bien distinctes dans le processus de découpage et de collage. Ce sont par ordre d'entrée en scène :

1°) *Le cadrage sur la portion d'image à découper* (voir figure 9.4a).

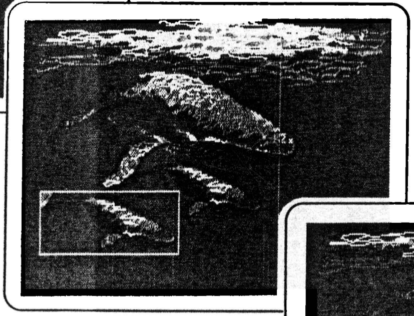
2°) *La découpe de la portion d'image cadrée* (en réalité, la découpe ressemble plus à une photographie dans la mesure où l'on ne détruit pas l'image d'origine).

3°) *Le déplacement de la découpe* (voir figure 9.4 b)

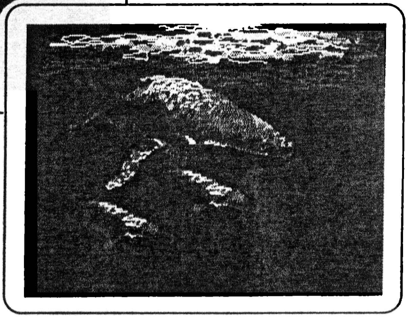
4°) *Le collage de la découpe* à son nouvel emplacement sur la même image ou sur une autre image (voir figure 9.4c).



◀ (a) Cadrage et découpe d'une portion d'image



(b) Déplacement de la découpe ▶



(c) Collage de la découpe ▶

Fig. 9.4 : Mise en œuvre du découpage et collage d'image

Ceci étant, il vous faut taper la première partie du programme que voici en ayant pris le soin de vous procurer une image (allez donc découper du vide...). En l'occurence, il s'agit de la photographie d'une baleine et de son baleineau évoluant sur le dos sous la surface de l'eau(*), le jeu constituant à rajouter un second baleineau au premier sans que la mère s'en aperçoive :

```
10 DIM DECOUPE(4000),FOND(4000)
20 KEY OFF:SCREEN 1:COLOR 8,1:CLS
30 BLOAD"BALEINES.IMA"
40 X=0:Y=0:H=7:V=4
50 GET(X,Y)-(X+H,Y+V),DECOUPE
60 LINE(X,Y)-(X+H,Y+V),,B
70 TOUCHE$=INKEY$:IF TOUCHE$="" THEN 70
80 ASCII=ASC(TOUCHE$):IF ASCII=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1))
90 PUT(X,Y),DECOUPE,PSET
100 IF ASCII=71 AND V>4 THEN V=V-1
110 IF ASCII=72 AND Y>0 THEN Y=Y-1
120 IF ASCII=73 AND X+H<319 THEN H=H+1
130 IF ASCII=75 AND X>0 THEN X=X-1
140 IF ASCII=77 AND X+H<319 THEN X=X+1
150 IF ASCII=79 AND H>7 THEN H=H-1
160 IF ASCII=80 AND Y+V<199 THEN Y=Y+1
170 IF ASCII=81 AND Y+V<199 THEN V=V+1
180 IF ASCII<>32 THEN 50
```

Cadrage... (lignes 100 à 170)

... et découpe de la portion d'image ! (ligne 180)

Cette première partie de programme étant quasi-identique à celle de la loupe électronique, hormis les fonctions d'agrandissement (touches [Pg Up] et [Pg Dn]) et de réduction (touches [Home] et [End]) des dimensions du cadre, il n'y a pas lieu de s'y attarder plus longtemps que celà ne le mérite. Notez quand même, que c'est l'enfoncement de la barre d'espacement qui réalise la découpe de la portion d'image (ligne 180).

La découpe étant faite, il vous reste à la déplacer et à la coller à son nouvel emplacement, ce que réalise fort bien la seconde partie du programme que voici :

```
190 GET(X,Y)-(X+H,Y+V),FOND
200 PUT(X,Y),DECOUPE,PSET:LINE(X,Y)-(X+H,Y+V),,B
210 TOUCHE$=INKEY$:IF TOUCHE$="" THEN 210
220 ASCII=ASC(TOUCHE$):IF ASCII=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1))
230 PUT(X,Y),FOND,PSET
240 IF ASCII=72 AND Y>0 THEN Y=Y-1
250 IF ASCII=75 AND X>0 THEN X=X-1
260 IF ASCII=77 AND X+H<319 THEN X=X+1
270 IF ASCII=80 AND Y+V<199 THEN Y=Y+1
280 IF ASCII<>32 THEN 190
290 PUT(X,Y),DECOUPE,PSET
300 GOTO 40
```

Déplacement... (lignes 240 à 270)

... et collage de la découpe ! (lignes 280 et 290)

(*) Cette image est extraite de la bibliothèque d'images fournie avec le logiciel de dessin PC PAINT.

PUT option PSET

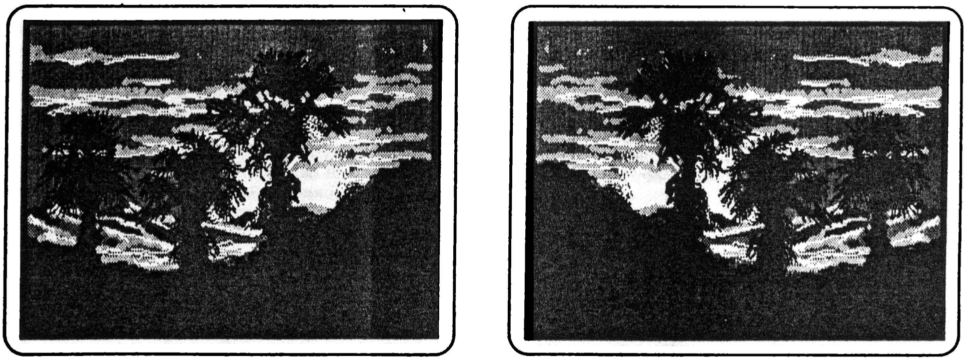
Que dire sur cette seconde partie, si ce n'est qu'elle ressemble comme deux gouttes d'eau à la première. Tout au plus, pouvez vous noter que l'on redesine le FOND par dessus la DECOUPE de façon à l'effacer (ligne 230) en se servant de l'option PSET de l'instruction PUT. Sans cela, on aurait dessiné les pixels du FOND dans la couleur complémentaire de ceux de la DÉCOUPE (vous parlez d'un gâchis...). Quant au collage final, il s'obtient tout simplement avec l'enfoncement de la barre d'espacement (lignes 280 et 290).

Le découpage et le collage de portions d'images sont réalisables sous Basic grâce aux instructions graphiques GET et PUT.

L'EFFET MIROIR

Je ne sais si vous êtes comme moi, mais je ne manque jamais une occasion capable de me faire économiser des heures de travail, surtout quand il s'agit de dessiner à la loupe une image qui présente un, voire deux axes de symétrie. En effet, pourquoi perdre du temps à dessiner plusieurs fois ce qui pourrait être obtenu en quelques secondes grâce à l'effet « miroir » !

Imaginant qu'il en est de même en ce qui vous concerne, je vous propose d'analyser en détail le processus qui permet de reproduire l'effet d'un miroir sur tout ou partie d'une image. Cette fois-ci, c'est l'image d'un coucher de soleil (image extraite de la bibliothèque d'images fournie avec le logiciel de dessin PC PAINT) qui va servir à nos expériences de réflexion.



(a) Image d'origine (b) Image réfléchie

Fig. 9.5 : Réflexion d'une image dans un miroir vertical.

Examinons pour débuter la réflexion de la totalité de l'image du coucher de soleil sur un miroir vertical (voir figures 9.5a et 9.5b). Comment aboutit-on à une telle image réfléchie ?... Mais en analysant l'image colonne par colonne et en permutant les colonnes de **GAUCHE** et de **DROITE** au fur et à mesure de leur analyse, comme dans l'exemple que voici :

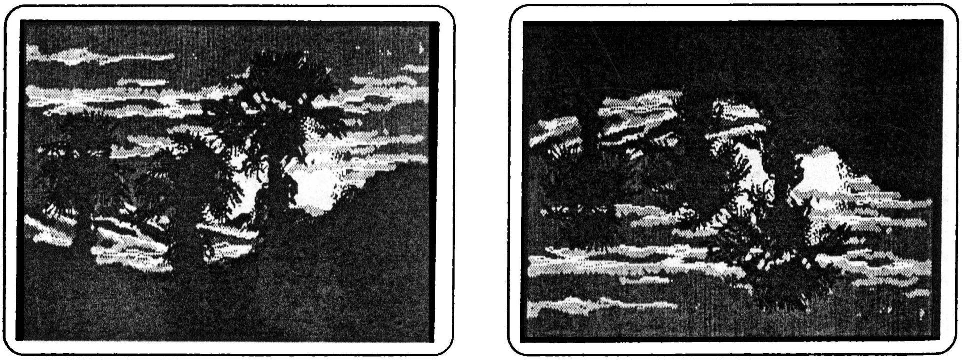
Analyse des colonnes de gauche et de droite de l'image ...

```
10 DIM GAUCHE(50),DROITE(50)
20 KEY OFF:SCREEN 1:COLOR 8,0:CLS
30 BLOAD"COUCHER.IMA"
40 FOR X=0 TO 159
50 GET(X,0)-(X,199),GAUCHE
60 GET(319-X,0)-(319-X,199),DROITE
70 PUT(X,Y),DROITE,PSET
80 PUT(319-X,Y),GAUCHE,PSET
90 NEXT
```

... et leur permutation pour obtenir l'effet miroir désiré !

A votre avis, peut-on faire plus simple ?... Après chargement de l'image (ligne 30), une boucle analyse deux à deux les 160 colonnes de chaque demi-écran (ligne 40), en mémorisant à chaque fois le couple de colonnes de **GAUCHE** et de **DROITE** dans 2 tableaux (lignes 50 et 60) avant de les dessiner l'une par dessus l'autre, en permutant celle de **GAUCHE** à **DROITE** et vice-versa (lignes 70 et 80).

Le principe de l'effet miroir étant désormais acquis (du moins je l'espère...), voici comment produire une image réfléchie dans un miroir horizontal (voir figures 9.6a et 9.6b) :



(a) Image d'origine **(b) Image réfléchie**

Fig. 9.6 : Réflexion d'une image dans un miroir horizontal

Analyse des lignes du haut et du bas de l'image...

```
10 DIM HAUT(20),BAS(20)
20 KEY OFF:SCREEN 1:COLOR 8,0:CLS
30 BLOAD"COUCHER.IMA"
40 FOR Y=0 TO 99
50 GET(0,Y)-(319,Y),HAUT
60 GET(0,199-Y)-(319,199-Y),BAS
70 PUT(0,Y),BAS,PSET
80 PUT(0,199-Y),HAUT,PSET
90 NEXT
```

... et leur permutation pour obtenir l'effet miroir désiré !

Bien que ce second exemple, calqué sur le premier, puisse vous paraître superflu, je ne puis que vous en conseiller la frappe et l'exécution. Vous découvrirez à travers lui que la machine que vous avez entre les mains traite de façon beaucoup plus rapide les tableaux à lignes horizontales que ceux à colonnes verticales, ce qui est confirmé par le fait qu'une ligne de 320 pixels ne nécessite qu'un tableau à 20 indices (ligne 10), tandis qu'une colonne de 200 pixels en nécessite un de 50 indices.

Il conviendra donc d'être vigilant sur cette différence de vitesse de traitement (mémorisation et affichage) des tableaux graphiques selon qu'ils sont à prédominance horizontale ou verticale. C'est d'ailleurs pourquoi vous ne trouverez dans ce chapitre aucun exemple de défilement horizontal d'une image, comme cela se produit lorsque l'on cherche dans une animation à faire bouger le décor de fond. C'est uniquement dû au temps anormalement long qu'un tel défilement horizontal nécessite. Pensez, plus de 8 minutes pour une image entière !!!

L'effet miroir sur une image s'obtient en permutant deux à deux les colonnes ou les lignes situées à égale distance de l'axe de symétrie.

CHARGEMENT ET SAUVEGARDE D'UNE IMAGE

BLOAD

Le chargement d'une image est une opération que vous avez déjà eu l'occasion d'exécuter à plusieurs reprises tout au long de ce chapitre. Il suffit de taper un **BLOAD** (*) suivi du nom de fichier de l'image (placé entre guillemets...) et le tour est joué. Seule petite précaution à prendre : être en mode graphique avant de charger l'image sinon, c'est peine perdue !

Le chargement d'une image graphique est simple à mettre en œuvre, mais en est-il de même avec la sauvegarde ?... Là, on se doit de nuancer la réponse selon les circonstances !

Imaginez que vous souhaitiez sauvegarder une image aprés lui avoir fait subir un traitement comme par exemple après avoir agrandi la famille de Dame Baleine. Comment allez-vous procéder ?

(*) La fiche détaillée de l'instruction BLOAD figure à la page 193 du volume 2 de Pratique des IBM-PC et compatibles.

Une première méthode (assez primaire, il faut bien l'avouer...) consiste à insérer dans le programme Basic, à un endroit adéquat (généralement en fin de traitement...) les deux lignes d'instructions que voici :

Çà, c'est l'adresse de début de l'image...

```
100 DEF SEG=&HB800
110 BSAVE"IMAGE",0,&H4000
```

... tandis que ceci correspond à sa longueur !

A quoi correspondent toutes ces indications ?... Pour la ligne 100, il s'agit de définir l'adresse de début de la zone-mémoire correspondant à l'écran graphique. Or, celui-ci débute en B800 (en hexadécimal) ou 47104 (en décimal). Fort de cette précision *indispensable*, la ligne suivante effectue la sauvegarde de l'image à l'aide de l'instruction **BSAVE** (*).

BSAVE

Notez que la syntaxe de cette instruction nécessite que lui soit précisée, après un 0 qui correspond au décalage, la longueur de la zone-mémoire de l'écran graphique qui est de 16K (quel que soit le mode graphique employé), ce qui correspond à 4000 (en hexadécimal) ou 16384 (en décimal).

Tout ceci est très bien, me direz-vous, mais relève plus du bricolage que d'une méthode rationnelle de sauvegarde !... Cela est vrai et c'est pourquoi il vous est présenté une seconde méthode qui, sans abîmer ou altérer l'image, vous permet de sauvegarder cette dernière quand bon vous semble et sous le nom qui vous convient, comme dans l'exemple que voici :

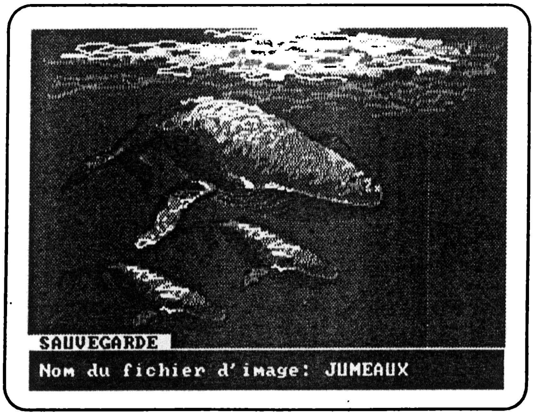


Fig. 9.7 : Création d'une fenêtre de texte dans l'image.

Or, comme vous pouvez le constater, cette méthode de sauvegarde passe par la création d'une fenêtre de texte dans le bas de l'image, fenêtre que l'on met à profit pour spécifier le nom de fichier sous lequel on veut sauvegarder l'image.

(*) La fiche détaillée de l'instruction figure à la page 194 du volume 2 de Pratique des IBM-PC.

Et comment crée-t-on une telle fenêtre ?... Mais avec trois fois rien, comme l'illustre le petit programme de démonstration que voici :

Après qu'elle ait été dimensionnée...

... cette fenêtre de texte apparaît...

... ou disparaît selon votre bon vouloir !

```
10 DIM FENETRE(640),SAUVEGARDE(48)
20 KEY OFF:SCREEN 1:COLOR 8,1:CLS
30 LOCATE 2,2:PRINT"SAUVEGARDE"
40 GET(0,7)-(95,14),SAUVEGARDE
50 CLS:BLOAD"BALEINES.IMA"
60 TOUCHE$=INKEY$:IF TOUCHE$="" THEN 60
70 GET(0,168)-(319,199),FENETRE
80 PUT(0,168),SAUVEGARDE,PRESET
90 LINE(0,176)-(319,199),,B:LINE(1,177)-(318,198),0,BF
100 LOCATE 24,2:INPUT;"Nom du fichier d'image: ",FICHIER$
110 PUT(0,168),FENETRE,PSET
120 DEF SEG=&HB800:BSAVE FICHIER$+".IMA",0,&H4000
```

Hormis la ligne 10 de dimensionnement de la fenêtre, seules les lignes 70 à 120 servent à établir et à effacer cette dernière. Notez l'emploi du point-virgule (;) à la suite de l'instruction INPUT (ligne 100) !... A quoi peut-il bien servir ?... Mais à éviter le terrible saut de ligne que l'ordinateur a pour habitude d'exécuter aprés qu'on ait appuyé sur la touche [RETURN] et qui aurait eu pour conséquence de pousser l'image vers le haut de l'écran en la déchirant vers le bas.

Avant de sauvegarder une image avec l'instruction BSAVE, il y a lieu de définir au préalable l'adresse de début (&HB800) de la zone-mémoire de l'écran graphique.

RECOPIE DES ÉCRANS GRAPHIQUES

Comme vous le savez pour l'avoir maintes fois employé, l'IBM-PC (ou ses compatibles) que vous avez entre les mains est dôté d'une touche qui permet d'obtenir instantanément la recopie sur papier du contenu de l'écran. Cette touche (ou plutôt ces deux touches devrait-on dire...), c'est [⇧] — [PrtSc].

GRAPHICS

Toutefois, l'emploi de cette touche est limité à la recopie des écrans textes si vous l'utilisez telle quelle !... Pour obtenir la recopie des écrans graphiques, il vous faut charger au préalable le fichier GRAPHICS qui figure sur la disquette du DOS.

SYSTEM

Pour ce faire, il vous faut passer sous le contrôle du DOS en tapant la commande SYSTEM. Ce faisant, vous obtenez le caractère d'appel du DOS (c'est le signe plus grand que) à la suite duquel vous tapez le nom du fichier GRAPHICS, comme indiqué ci-dessous :

Sans ce fichier, il n'y a pas de recopie d'écrans graphiques possible !

```
A>GRAPHICS
```

Une autre solution, plus élégante au demeurant (surtout quand on se rend compte en appuyant sur [↑] – [PrtSc] que le fichier **GRAPHICS** n'a pas été chargé...), consiste à charger celui-ci à l'aide d'un fichier à auto-exécution lors de l'amorçage de l'ordinateur. Cela ne peut que libérer l'esprit et faire gagner du temps !

Comment crée-t-on un fichier à auto-exécution ?... Mais en passant d'abord sous DOS en tapant la commande **SYSTEM**, puis en écrivant à la suite du caractère d'appel la séquence que voici :

```
A:\>COPY CON:AUTOEXEC.BAT
ECHO OFF
KEYBFR
GRAPHICS
BASICA
^Z
```

Grâce au fichier à auto-exécution, le fichier GRAPHICS est chargé sans que l'on s'en aperçoive

Ainsi, le fichier **GRAPHICS** se trouve être chargé sans que l'on s'en aperçoive lors de la mise sous tension de l'ordinateur ou lors de son amorçage à chaud avec les touches [Ctrl] – [Alt] – [Del].

Remarque : *Bien que l'ordre de chargement du fichier GRAPHICS importe peu, il est impératif que le fichier BASICA soit chargé en dernier.*

Ce point étant acquis, il vous reste à faire quelques essais de recopie des écrans graphiques. Ce faisant, vous découvrirez (avec surprise...) que la routine de recopie traite différemment les écrans à moyenne résolution de ceux à haute résolution. En effet, si les premiers sont recopiés dans le sens de la largeur de la feuille (comme les écrans textes...), il n'en est pas de même des seconds qui se présentent dans le sens de la longueur.

Mieux encore !... Amusez-vous à mesurer leurs dimensions. Vous découvrirez (avec stupeur cette fois-ci...) que les écrans à haute résolution sont les seuls à conserver sur papier le même rapport largeur/hauteur que sur l'écran. Fort de cette constatation, vous choisirez de préférence ce mode de recopie, quitte pour cela à transférer vos écrans couleurs en moyenne résolution en écrans monochromes à haute résolution. C'est la garantie d'une recopie d'écran correcte !

Mais ce n'est pas tout !... Il vous faut encore savoir que cette recopie d'écran s'effectue en vidéo inversée, c'est-à-dire que ce qui est lumineux à l'écran devient noir sur le papier, et vice-versa. Or, si une telle reproduction est parfaite pour du texte qui apparaît lumineux sur fond noir à l'écran et noir sur fond blanc sur le papier, il n'en est pas forcément de même en ce qui concerne les images, la preuve en est l'image que voici : (cette image est extraite de la bibliothèque d'images livrée avec le logiciel de dessin PC PAINT).

(*) Pour de plus amples détails sur la procédure de création d'un fichier à auto-exécution, vous pouvez vous reporter à l'ouvrage de H. LILEN « Pratique des MS DOS et PC DOS » chez le même éditeur.

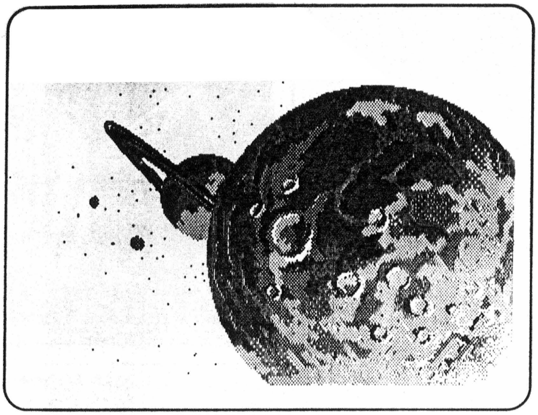


Fig. 9.8 : Recopie d'écran en vidéo inversée

Comme vous pouvez le constater de visu, une telle recopie d'écran se doit d'être faite sans inversion vidéo car, jusqu'à nouvel ordre, l'espace est noir !... C'est pourquoi il faut faire subir à l'image une première inversion vidéo sur l'écran, la recopie se chargeant alors de redresser la situation telle qu'elle était avant la première inversion.

PUT option PRESET

Et comment inverse-t-on une image sur l'écran ?... Mais en usant de l'option **PRESET** de l'instruction **PUT**, comme c'est le cas dans l'exemple que voici :

L'option PRESET de l'instruction PUT sert à inverser une image vidéo

```
10 DIM ECRAN(4000)
20 KEY OFF:SCREEN 2:CLS
30 BLOAD"ESPACE.IMA"
40 GET(0,0)-(639,199),ECRAN
50 PUT(0,0),ECRAN,PRESET
```

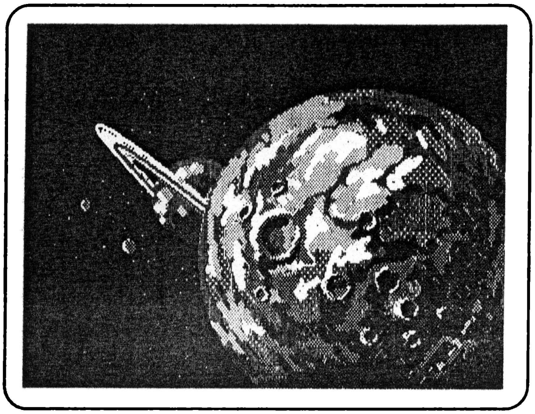
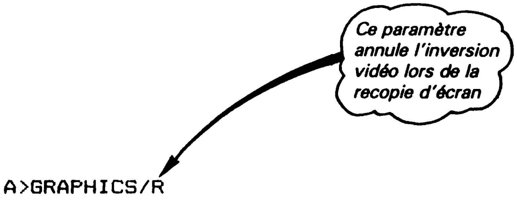


Fig.9.9 : Recopie d'écran correcte

Remarque : *Si vous disposez d'un DOS récent (version 3.1 ou supérieure...), vous pouvez obtenir directement l'inversion vidéo lors de la recopie d'écran en spécifiant le paramètre/R à la suite du fichier GRAPHICS.*



La recopie des écrans graphiques n'est possible qu'à la condition que le fichier GRAPHICS ait été préalablement chargé.

Exercices sur le chapitre 9

1. Quel est le rôle de l'instruction SCREEN quand elle est associée à l'instruction WINDOW ?

   ☐ Elle permet de passer d'un mode graphique à un autre sans perturber les limites d'écran fixées par l'instruction WINDOW.
   ☐ Elle inverse l'orientation de l'axe vertical de l'écran graphique.

2. Le sous-langage DRAW est-il dépendant des limites d'écran ?

   ☐ Oui, au même titre que les autres instructions graphiques
   ☐ Non, seules le sont les instructions LINE et CIRCLE.

3. A quoi sert l'option OR de l'instruction PUT ?

   ☐ A dessiner une image à la place d'une autre.
   ☐ A mélanger deux images.

4. Est-il indispensable de spécifier l'adresse de début de la zone-mémoire de l'écran graphique lors d'une sauvegarde ?

   ☐ Oui, sans quoi on sauvegarde 16 koctets de n'importe quoi.
   ☐ Non, si cela a déjà été fait précédemment.

5. Comment obtient-on une image vidéo inversée ?

   ☐ en l'imprimant sur papier.
   ☐ en employant l'option XOR (c'est l'option par défaut) de l'instruction PUT.
   ☐ en employant l'option PSET de l'instruction PUT.
   ☐ en rajoutant le paramètre /R à la suite du nom du fichier GRAPHICS lors de son chargement.

Réponses page 174.

Solution des exercices sur le chapitre 8

1. Parce que c'est la seule disposition possible lorsque l'animation porte sur plusieurs formes.
2. Non, car elle est incapable de dessiner dans la couleur complémentaire du fond.
3. Oui, dans tous les cas !
4. L'ordinateur émet un message d'erreur.
5. Parce qu'il correspond au code des touches du clavier numérique.
6. Il n'y en a pas !

Solution des exercices sur le chapitre 9

1. Elle inverse l'orientation de l'axe vertical de l'écran graphique.
2. Non, seules les instructions LINE et CIRCLE le sont.
3. A mélanger deux images.
4. Oui, sans quoi on sauvegarde 16 koctets de n'importe quoi ET Non, si cela a déjà été fait précédemment, à la condition toutefois, qu'aucune autre adresse n'ait été définie entre-temps à l'aide de l'instruction DEF SEG.
5. En l'imprimant sur papier. En ce qui concerne l'instruction PUT, c'est l'option PRESET (non mentionnée) qui permet d'obtenir l'inversion vidéo à l'écran.

CHAPITRE X

LA SOURIS MICROSOFT

Vous disposez d'une souris et vous ne savez pas la programmer en mode graphique ?... Qu'à cela ne tienne !... Le présent chapitre va combler cette lacune en ce sens qu'il lui est entièrement consacré. Vous y découvrirez que déplacer une cible sur l'écran, dessiner à main levée, sélectionner une option dans un menu, même si celui-ci est du type « déroulant », ou encore mettre en place des icônes sur l'écran est tout simlement l'enfance de l'art.

Menu du chapitre 10

Activation de la souris
Fonctions de programmation
Changement de curseur mobile
Dessin à main levée
Choix d'options dans un menu
Menus déroulants
Déplacement d'icônes sur l'écran

Quel est votre choix : ■

ACTIVATION DE LA SOURIS

Puisque le graphisme vous intéresse et que vous disposez d'une souris, pourquoi ne pas l'incorporer dans vos programmes personnels ?... Si parfois son utilité est contestable avec les logiciels de type traitement de texte, tableur ou gestion de fichiers (on a aussi vite fait de taper sur le clavier...), il n'en est pas de même en ce qui concerne les logiciels graphiques. Véritable prolongement de la main, elle permet alors d'accéder instantanément à n'importe quel point précis de l'écran, ce qui ne peut être obtenu au clavier qu'au travers d'un nombre excessif de manipulations des touches fléchées (sans parler du temps que cela prend...).

MOUSE

Si donc vous êtes convaincus de son utilité (dans le cas contraire, sautez directement au chapitre suivant...), il ne vous reste plus qu'à la rendre bien obéissante... cette petite souris !... Pour ce faire, il vous faut commencer par charger son logiciel de commande dénommé **MOUSE**.

Comment fait-on pour charger le logiciel MOUSE ? On commence par taper la commande SYSTEM qui vous permet d'accéder directement au MS DOS. Ceci fait, vous insérez la disquette de la souris dans le lecteur A (celui de gauche) à la place de votre disquette de travail et vous tapez :

Sans le fichier MOUSE, la souris demeure sans vie

```
A>MOUSE
```

Il est bien entendu qu'un tel chargement manuel du fichier MOUSE de la souris n'a de sens que dans la mesure où vous souhaitez procéder à quelques essais (sans suite...) de programmation de celle-ci. Dans le cas contraire, c'est-à-dire si vous envisagez d'incorporer la souris à l'un de vos programmes, il y a lieu de procéder à un chargement automatique du fichier MOUSE au travers d'un AUTOEXEC. BAT (*).

Pour cela, il vous faut tout d'abord copier le fichier MOUSE. COM sur votre disquette de travail en plaçant celle-ci dans le lecteur B (celui de droite), la disquette de la souris étant dans le lecteur A (l'autre...). Vous tapez alors :

Copie du fichier MOUSE.COM sur votre disquette de travail

```
A>COPY MOUSE.COM B:
```

Ceci fait, il ne vous reste plus qu'à procéder à l'écriture du fichier à autoexécution en tapant :

Chargement automatique du fichier MOUSE !

```
A>COPY CON:AUTOEXEC.BAT
KEYBFR
GRAPHICS
MOUSE
BASICA
^Z
```

Vous noterez que le chargement du fichier MOUSE s'effectue avant celui du Basic (BASICA). Si une telle disposition est impérative, le fichier BASICA devant être chargé en dernier, rien par contre n'empêche d'intervertir l'ordre de chargement des trois premiers fichiers.

Le fichier de la souris étant maintenant chargé, il ne reste plus qu'à localiser sa position dans la mémoire à l'aide de quelques PEEK bien placés, tels que ceux des lignes 30 et 40 :

Localisation du fichier MOUSE dans la mémoire...

```
10 KEY OFF:SCREEN 2:CLS
20 DEF SEG=0
30 SEGMENT=256*PEEK(207)+PEEK(206)
40 SOURIS=256*PEEK(205)+PEEK(204)+2
50 DEF SEG=SEGMENT
```

(*) Si cette procédure vous est inconnue, vous pouvez vous reporter avec profit à l'ouvrage Pratique des MS DOS et PC DOS de H. Lilen chez le même éditeur.

... et à procéder à son activation à l'aide de la ligne 60 que voici :

```
60 M1%=0:CALL SOURIS(M1%,M2%,M3%,M4%)
```

Cette ligne un peu spéciale appelle quelques commentaires !... Tout d'abord, il importe de la débuter par un M1 % = 0. Comme vous le verrez dans les pages suivantes, M1 % n'est pas une variable ordinaire. C'est en fait une sorte d'instruction de programmation de la souris. Il n'est donc pas question de chercher à vouloir en changer !... Il en est presque de même en ce qui concerne les 4 variables (en fait ce sont des attributs...) placées entre parenthèses. Si leur présence est **obligatoire**, seule celle qui correspond à la fonction d'entrée se doit de porter la même appellation. Pour les autres, toutes les libertés sont permises !

Remarque : *Il est possible d'éviter l'emploi du signe % qui caractérise individuellement chaque variable entière en les rendant toutes entières dès le début du programme par un DEFINT A-Z.*

Ceci dit, il est temps de passer à l'exécution de ce micro-programme !... Normalement (mais sait-on jamais avec une souris...), vous devriez voir apparaître le bout de son museau dans un coin de l'écran :

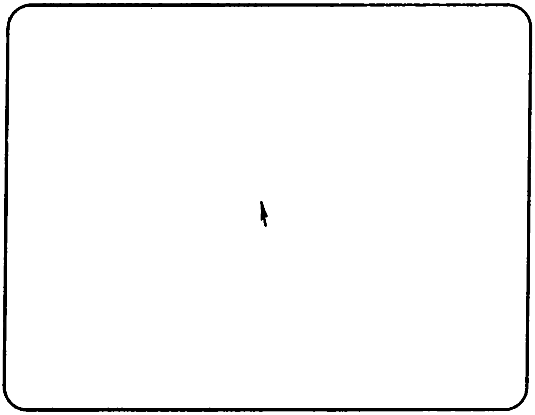


Fig. 10.1 : La souris montre le bout de son museau

La souris Microsoft ne peut fonctionner que si le fichier MOUSE est chargée en mémoire et activé par un CALL SOURIS.

FONCTIONS DE PROGRAMMATION

Voyons maintenant d'un peu plus près les fonctions de programmation de la souris. Celles-ci au nombre de 16 permettent de fixer les divers paramètres d'entrée du logiciel de commande de la souris, tels que vitese de déplacement, dimensions de la fenêtre, allumage et extinction du curseur, etc, etc... D'autre part, certaines fonctions sont spécialisées dans la collecte des données en provenance de la souris (position du curseur, état des boutons-poussoirs, nombre de cliquetages, etc, etc...).

Voici d'ailleurs à ce sujet un tableau récapitulatif des fonctions les plus usitées(*) :

Fonction	Paramètres d'entrée	Paramètres de sortie
0	*Installation par défaut de la souris :* M1 = 0	M1 = Etat de la souris M2 = Nombre de boutons (toujours égal à 2)
	L'état de la souris vaut 1 si la souris est installée et le fichier MOUSE chargé, 0 dans le cas contraire. La fonction 0 remet à zéro le logiciel de commande de la souris et installe par défaut les paramètres suivants : Etat du curseur : visible Position du curseur : centre de l'écran Type de curseur : flèche Position relative du point de référence : (−1, −1) Limites horizontales de déplacement : 0,639 Limites verticales de déplacement : 0,199 Vitesse de déplacement (en mickeys/8 pixels) : 8 (horizontal), 16 (vertical)	
1	*Allumage du curseur :* M1 = 1	
	La fonction 1 rend visible le curseur après qu'il ait été effacé par la fonction 2.	
2	*Extinction du curseur :* M1 = 2	
	La fonction 2 rend le curseur invisible. Il convient d'utiliser cette fonction avant toute modification de la portion d'image située sous le curseur, sans quoi elle peut être altérée.	
3	*Position du curseur et état des boutons-poussoirs :* M1 = 3	M2 = Etat des boutons M3 = Position horizontale du curseur M4 = Position verticale du curseur
	La fonction 3 délivre la position du curseur et l'état des boutons-poussoirs. Le paramètre M2 vaut 0 si aucun bouton n'est enfoncé, 1 si celui de gauche l'est, 2 si celui de droit l'est et 3 si les deux le sont en même temps.	
4	*Positionnement du curseur :* M1 = 4 M3 = Nouvelle position horizontale M4 = Nouvelle position verticale	
	La fonction 4 positionne le curseur à l'intérieur des limites définies par les fonctions 7 et 8.	

(*) Pour les autres, veuillez vous reporter à la documentation fournie avec la souris.

7	*Fixation des limites horizontales de déplacement du curseur :* M1 = 7 M3 = limite de gauche M4 = limite de droite	
	La fonction 7 fixe les limites horizontales de déplacement du curseur. Notez que quel que soit le mode graphique employé, l'écran est considéré comme possédant 640 pixels.	
8	*Fixation des limites verticales de déplacement du curseur :* M1 = 8 M3 = limite haute M4 = limite basse	
	La fonction 8 fixe les limites verticales de déplacement du curseur. Notez que si vous intervertissez les limites haute et basse, le logiciel les rétablit de lui-même.	
9	*Définition du curseur :* M1 = 9 M2 = Position relative horizontale du point de référence M3 = Position relative verticale du point de référence M4 = Masques de fond d'écran et du curseur	
	La fonction 9 définit la forme du curseur ainsi que la position relative de son point de référence.	
15	*Vitesse de déplacement du curseur :* M1 = 15 M3 = Nombre de mickeys/8 pixels (axe horizontal) M4 = Nombre de mickeys/8 pixels (axe vertical)	
	La fonction 15 définit la vitesse de déplacement du curseur en mickeys/8 pixels. Par défaut, celle-ci est de 8 mickeys/8 pixels sur l'axe horizontal et de 16 mickeys/8 pixels sur l'axe vertical.	

Assez de tableaux !... Vite un exemple !... En voici un qui, tout en employant le curseur défini par défaut (la flèche), affiche celle-ci dans l'angle inférieur droit de l'écran, limite ses déplacements à 1/4 de l'écran et lui confère une vitesse de tortue (pour une souris c'est un comble...) :

DEFINT A-Z simplifie l'écriture des fonctions de programmation

```
10 DEFINT A-Z
20 DEF SEG=0:SEGMENT=256*PEEK(207)+PEEK(206)
30 SOURIS=256*PEEK(205)+PEEK(204)+2:DEF SEG=SEGMENT
40 KEY OFF:SCREEN 2:CLS
50 M1=0:CALL SOURIS(M1,M2,M3,M4)
60 M1=2:CALL SOURIS(M1,M2,M3,M4)
70 M1=7:M3=320:M4=639:CALL SOURIS(M1,M2,M3,M4)
80 M1=8:M3=100:M4=199:CALL SOURIS(M1,M2,M3,M4)
90 M1=15:M3=80:M4=160:CALL SOURIS(M1,M2,M3,M4)
100 M1=4:M3=560:M4=150:CALL SOURIS(M1,M2,M3,M4)
110 M1=1:CALL SOURIS(M1,M2,M3,M4)
```

Définition des paramètres de la souris à l'aide des fonctions de programmation

Qui fait quoi dans tout cela ?... La ligne 50 installe par défaut la souris (en l'occurence, cette ligne n'a d'autre but que de définir par défaut la flèche servant de curseur). Ceci fait, la ligne 60 l'efface et, tandis qu'elle est invisible, les lignes 70 à 100 déterminent les limites de sa fenêtre de déplacement (320, 639, 100, 199), sa vitesse de tortue (80 et 160) et sa nouvelle position sur l'écran (560, 150). Quant à sa réapparition sur l'écran, c'est la ligne 110 qui en est la cause !

Les fonctions de programmation servent à définir les paramètres de fonctionnement de la souris.

CHANGEMENT DE CURSEUR MOBILE

Le curseur en forme de flèche vous convient ?... Oui ?... Alors tant mieux !... Non ?... Dans ce cas, il faut vous prendre par la main et définir votre propre curseur mobile.

Comment fait-on pour définir son propre curseur ?... Ça, c'est une longue histoire que je vais vous conter !... Tout d'abord, il vous faut savoir que vous disposez de 16 × 16 éléments binaires pour définir la forme du curseur. Donc, pas question de dessiner un éléphant, l'exéguïté du lieu ne le permettrait pas !...

Ensuite, il vous faut savoir qu'à chaque pixel du mode graphique à haute résolution correspond un élément binaire et un seul, tandis qu'il en correspond deux dans le mode graphique à moyenne résolution, chaque groupe de deux éléments binaires définissant la couleur parmi 4 du pixel en question. En clair, cela revient à dire que le curseur que l'on peut dessiner en moyenne résolution ne peut dépasser 8 pixels de large sur 16 pixels de haut, tandis qu'il peut atteindre une taille de 16 × 16 pixels en haute résolution.

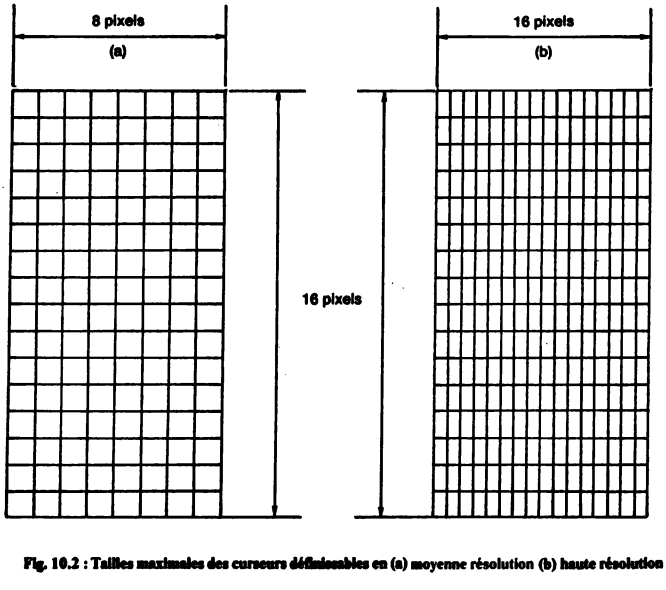


Fig. 10.2 : Tailles maximales des curseurs définissables en (a) moyenne résolution (b) haute résolution

Mais ce n'est pas tout !... Il faut que je vous parle du masque de fond d'écran situé sous le curseur !... Comme je suppose que vous allez vouloir déplacer le curseur sur un fond d'écran sans abîmer ce dernier, il va vous falloir constituer un masque de fond d'écran composé entièrement de « 1 ». Et quelle doit être la taille de ce masque de fond d'écran ?... Celle-ci se doit d'être de 16 × 16 éléments binaires, c'est-à-dire qu'elle doit correspondre à la totalité des pixels disponibles pour le curseur...

Voyons tout cela en détail avec un petit exemple !... Imaginez que vous ayez besoin d'un curseur en forme de cible et que, travaillant en moyenne résolution, celle-ci doit être de couleur violette. Vous allez donc commencer par définir un tableau à 2 dimensions qui va obligatoirement devoir comporter le masque de fond d'écran placé en premier, suivi du masque du curseur. Or, sachant que le code de la couleur violette est 2 (10 en binaire), vous allez immanquablement aboutir aux deux masques que voici :

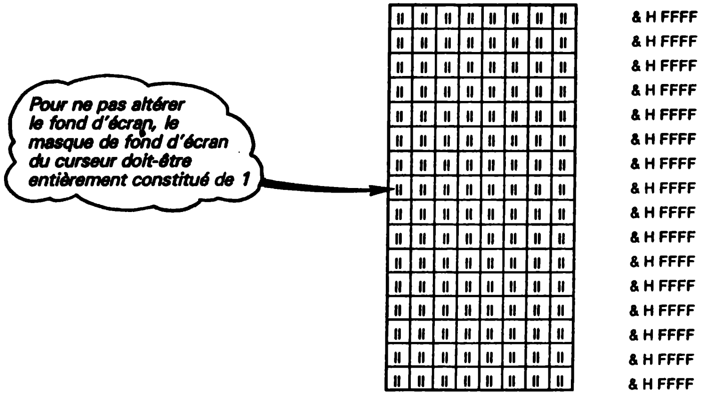


Fig. 10.3 : Masque de fond d'écran du curseur

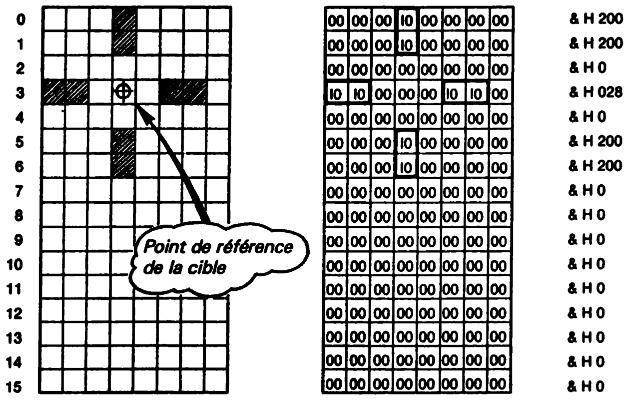


Fig. 10.4 : Masque du curseur

... qui à leur tour vont donner lieu au tableau à 2 dimensions **CURSEUR** que voilà :

Dimensionnement du tableau à 2 dimensions destiné à recevoir les 2 masques du curseur

```
10 DEFINT A-Z:DIM CURSEUR(15,1)
20 DEF SEG=0:SEGMENT=256*PEEK(207)+PEEK(206)
30 SOURIS=256*PEEK(205)+PEEK(204)+2:DEF SEG=SEGMENT
40 FOR I=0 TO 15:CURSEUR(I,0)=&HFFFF:NEXT
50 CURSEUR(0,1)=&H200
60 CURSEUR(1,1)=&H200
70 CURSEUR(3,1)=&HA028
80 CURSEUR(5,1)=&H200
90 CURSEUR(6,1)=&H200
```

Masque de fond d'écran...

... et masque du curseur !

Notez que le masque de fond d'écran est incorporé en premier dans le tableau (indice 0), tandis que celui du curseur l'est en second (indice 1).

Remarque : *L'instruction DIM ayant pour effet de remettre à zéro toutes les cases du tableau du CURSEUR, il est inutile de déclarer les lignes de valeur nulle du masque du curseur.*

Et maintenant qu'il est rempli, qu'est-ce qu'on fait de ce tableau ?... On l'introduit tout simplement dans le logiciel de la souris à l'aide de la fonction 9, comme ceci :

L'introduction du tableau CURSEUR se fait ici !

```
100 KEY OFF:SCREEN 1:COLOR 8,1:CLS
110 M1=9:M2=6:M3=3:CALL SOURIS(M1,M2,M3,CURSEUR(0,0))
120 M1=7:M3=0:M4=639:CALL SOURIS(M1,M2,M3,M4)
130 M1=8:M3=0:M4=199:CALL SOURIS(M1,M2,M3,M4)
140 M1=15:M3=8:M4=16:CALL SOURIS(M1,M2,M3,M4)
150 M1=4:M3=320:M4=100:CALL SOURIS(M1,M2,M3,M4)
160 M1=1:CALL SOURIS(M1,M2,M3,M4)
```

... en précisant pour celui-ci les indices de début (0, 0). Notez au passage que cette même ligne 110 définie également la position relative du point de référence du curseur. En l'occurence, il s'agit du point de coordonnée 6, 3. Pourquoi 6 en horizontal et non pas 3 ?... Parce que, rappelons-le les dimensions de l'écran sont de 640 × 200 unités (tout du moins du point de vue de la souris...) quel que soit le mode graphique employé. De ce fait, un décalage horizontal de 3 pixels en moyenne résolution correspond à un décalage de 6 unités pour la souris (pourquoi faire simple, quand on peut faire compliqué...)

Essayez donc le petit programme que vous venez de taper, juste histoire de vérifier que l'on ne vous raconte pas d'histoires !... Si la cible répond fidèlement aux sollicitations de la souris, dessinez donc quelques lignes en travers de l'écran pour vous assurer que le fond d'écran sur lequel passe la souris n'est pas altéré. Rien ne vaut de vérifier par soi-même !

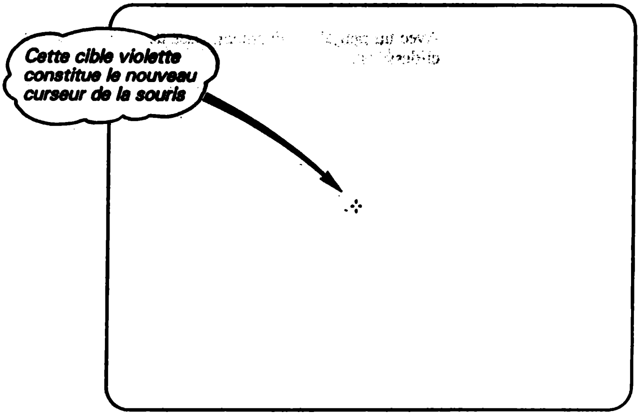


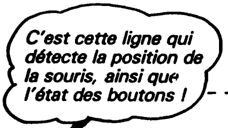
Fig. 10.5 : Curseur mobile définissable par l'utilisateur

La fonction 9 du logiciel de la souris permet de définir soi-même la forme et la couleur du curseur mobile.

DESSIN À MAIN LEVÉE

Cela vous direz de dessiner à main levée sur l'écran de votre ordinateur en vous servant de la souris comme d'un crayon ?... Oui ?... Alors suivez le guide !

Que vous faut-il pour cela ?... Une souris, un petit programme de 10 lignes et deux boutons !... Pourquoi deux boutons ?... Parce qu'il en faut un pour dessiner et un pour effacer !... Mais, trève de bavardage, essayez donc ce qui suit :




```
10 DEFINT A-Z
20 DEF SEG=0:SEGMENT=256*PEEK(207)+PEEK(206)
30 SOURIS=256*PEEK(205)+PEEK(204)+2:DEF SEG=SEGMENT
40 KEY OFF:SCREEN 2:CLS
50 LINE(0,0)-(639,199),,B
60 M1=0:CALL SOURIS(M1,M2,M3,M4)
70 M1=3:CALL SOURIS(M1,BOUTON,X,Y)
80 IF BOUTON=1 THEN LINE(X,Y)-(X+7,Y-3),1,BF
90 IF BOUTON=2 THEN LINE(X,Y)-(X+7,Y-3),0,BF
100 GOTO 70
```

Avec un peu d'entraînement, vous devriez être à même de faire mieux que l'exemple ci-dessous :

Fig. 10.6 : Un exemple de dessin à main levée.

Voyons un peu maintenant comment tout cela fonctionne !... Après la procédure habituelle d'activation de la souris (lignes 10 à 30), on passe en mode graphique à haute résolution (ligne 40) et on dessine un cadre tout autour de l'écran (ligne 50).

Ceci fait, on installe par défaut la souris (ligne 60) avec tout ce que cette procédure sous-entend (curseur en forme de flèche, position initiale au milieu de l'écran, déplacements autorisés sur la totalité de l'écran, vitesse de déplacement de 8 mickeys/8 pixels en horizontal et de 16 mickeys/8 pixels en vertical, etc, etc...)

Important : ***Avant tout appel des fonctions de programmation de la souris, il est indispensable d'avoir sélectionné au préalable le mode graphique à moyenne ou haute résolution selon le cas.***

Ensuite, c'est au tour de la ligne 70 (en fait, c'est la ligne capitale du programme) de détecter à la fois les coordonnées X, Y de la position du curseur sur l'écran et l'état des deux BOUTONs. Si vous appuyez sur le BOUTON de gauche, la ligne 80 se charge de vous dessiner un gros point noir de 8 X 4 pixels juste au-dessus du curseur. Si par contre, vous appuyez sur le BOUTON de droite (celui d'effacement), c'est la ligne 90 qui se charge d'effacer le même carré de 8 X 4 pixels situé au-dessus du curseur. Quant à la ligne 100, elle boucle sans fin sur la ligne 70 de façon à prendre en compte toute nouvelle position de la souris ou de changement d'état des boutons.

Pour dessiner à main levée, il suffit de détecter la position de la souris et l'état des boutons en se servant de la fonction 3.

CHOIX D'OPTIONS DANS UN MENU

Tout ce qui précédait n'étant qu'entrée en matière, passons maintenant aux choses sérieuses, à savoir la sélection d'options dans un menu !... Voici à titres de démonstration un exemple tout simple visant à obtenir le choix d'une option parmi six dans le menu du logiciel graphique que voici :

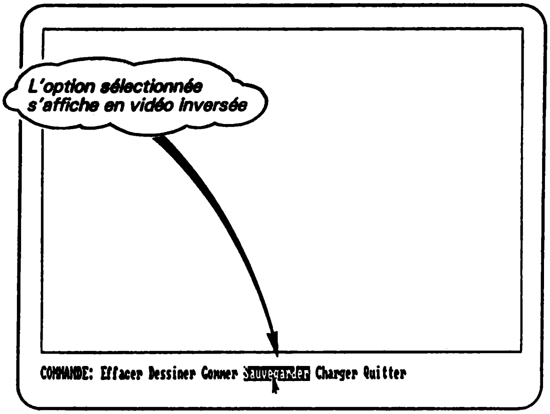
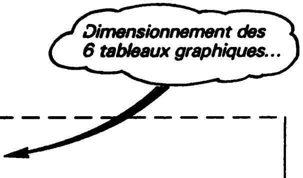


Fig. 10.7 : Choix d'options avec une souris

PUT option PRESET

Or, que nécessite un tel menu pour permettre le choix d'options avec une souris ?... Rien de bien particulier, si ce n'est de pouvoir afficher du texte en vidéo inversée sur l'écran graphique de manière à mettre en lumière l'option choisie. Et, paradoxe, c'est précisément là que réside toute la difficulté !... La solution proposée (il y en a d'autres...) consiste à placer chaque option dans un tableau graphique avec un GET et, ceci fait, à reproduire celle que l'on veut obtenir en vidéo inversée avec un PUT (*) utilisé en mode PRESET qui dessine, précisément, en vidéo inversée. Voici d'ailleurs ce que cela donne :




```
10 DEFINT A-Z:DIM EFFACER(33),DESSINER(33),GOMMER(25)
20 DIM SAUVEGARDER(45),CHARGER(33),QUITTER(33)
30 DEF SEG=0:SEGMENT=256*PEEK(207)+PEEK(206)
40 SOURIS=256*PEEK(205)+PEEK(204)+2:DEF SEG=SEGMENT
50 KEY OFF:SCREEN 2:CLS
60 LINE(0,0)-(639,176),,B
```

(*) La fiche détaillée de l'instruction PUT figure à la page 334 du volume 2 de Pratique des IBM-PC et compatibles.

```
70 COMMANDE$="COMMANDE: Effacer Dessiner Gommer Sauvegarder Charger Quitter"
80 LOCATE 24,1:PRINT COMMANDE$;
90 GET(80,184)-(135,191),EFFACER
100 GET(144,184)-(207,191),DESSINER
110 GET(216,184)-(263,191),GOMMER
120 GET(272,184)-(359,191),SAUVEGARDER
130 GET(368,184)-(423,191),CHARGER
140 GET(432,184)-(487,191),QUITTER
150 M1=0:CALL SOURIS(M1,M2,M3,M4)
160 M1=1:CALL SOURIS(M1,M2,M3,M4)
170 M1=3:CALL SOURIS(M1,BOUTON,X,Y):IF BOUTON<>1 THEN 170
180 IF Y<184 OR Y>191 OR X<76 OR X>=492 THEN 170
190 M1=2:CALL SOURIS(M1,M2,M3,M4)
200 LOCATE 24,1:PRINT COMMANDE$;
210 IF X>=76 AND X<140 THEN PUT(80,184),EFFACER,PRESET:CHOIX=1
220 IF X>=140 AND X<212 THEN PUT(144,184),DESSINER,PRESET:CHOIX=2
230 IF X>=212 AND X<268 THEN PUT(216,184),GOMMER,PRESET:CHOIX=3
240 IF X>=268 AND X<364 THEN PUT(272,184),SAUVEGARDER,PRESET:CHOIX=4
250 IF X>=364 AND X<428 THEN PUT(368,184),CHARGER,PRESET:CHOIX=5
260 IF X>=428 AND X<492 THEN PUT(432,184),QUITTER,PRESET:CHOIX=6
270 GOTO 160
```

... que l'on remplit ici...

... et que l'on affiche là !

Comme vous pouvez le constater, c'est la gestion des tableaux graphiques des noms d'option (dimensionnement, remplissage et affichage) qui occupe la plus grosse partie de ce programme. Pour le reste, c'est-à-dire le choix d'options dans un menu, ce sont les lignes 160 à 190 qu'il faut examiner de près.

La ligne 160 a pour objet de faire apparaître le curseur sur l'écran. Or, une telle ligne peut paraître inutile lorsque l'on sait que la fonction 0 rend déjà visible le curseur. En réalité, il faut se souvenir (voir fonction 2 de programmation) qu'il faut effacer le curseur avant toute modification du fond d'écran situé en-dessous de lui (ligne 190). Dans ces conditions, il y a lieu de le faire réapparaître après la dite modification et c'est ce à quoi s'emploit la ligne 160. Notez qu'une telle réapparition pourrait très bien s'effectuer à la ligne 270.

Ce point étant acquis, passons maintenant à la ligne 170. Seule la fin de cette ligne nous intéresse !... Que dit-elle ?... Que si l'on n'a pas appuyé sur le bouton de gauche (le seul qui soit pris en compte), on n'a pas le droit d'aller plus loin dans le programme. Pourquoi une telle écriture, vous demandez-vous ?... Mais pour éviter de pénétrer dans une portion de programme où l'on a rien à y faire (tout du moins tant que l'on ne clique pas la souris...).

On retrouve un barrage analogue à la ligne 180 où l'on rejète vers la ligne 170 toute position du curseur de la souris située en dehors de la ligne d'écriture des 6 options. Ce n'est que lorsque celui-ci est dans l'intervalle [184, 191] en Y (c'est l'équivalent graphique de la ligne 24 de texte...) et dans l'intervalle [76, 492] en X que l'on analyse en détail sa position avec les lignes 210 à 260. Notez l'attribution d'une valeur différente à la variable CHOIX selon l'option sur laquelle pointe le curseur.

Pour sélectionner une option dans un menu, il suffit de détecter la position de la souris après chaque cliquetage.

MENUS DÉROULANTS

Vous suivez !... Tant mieux parce que le précédent exercice de style nous ayant mis en appétit, nous voici prêt à aborder les menus de type « déroulants ». C'est quoi un menu déroulant ?... C'est un type de menu généralement associé à un logiciel de dessin qui, roulé sur lui-même, évite d'encombrer inutilement la place utile réversée au dessin. Ce n'est que lorsque l'on clique sur lui qu'il consent à se dérouler de façon à vous proposer les diverses options qu'il comporte, comme dans l'exemple « maison » que voici :

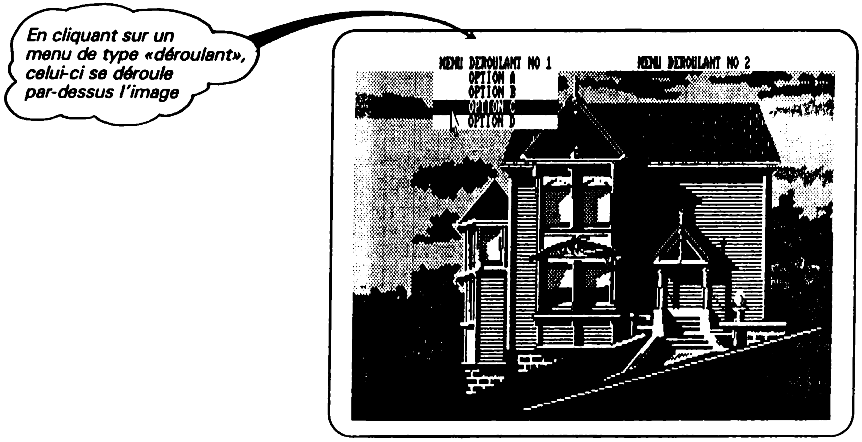


Fig. 10.8 : Un exemple de menu déroulant

Impressionnant, n'est-ce pas !... Or, que faut-il pour parvenir à un tel résultat ?... Bien peu de choses en vérité, puisque tout est contenu dans le petit programme que voici :

```
10 DEFINT A-Z:DIM HAUT(321),BANDE(321),IMAGE(873)
20 DIM OPTIONA(85),OPTIONB(85),OPTIONC(85),OPTIOND(85)
30 DEF SEG=0:SEGMENT=256*PEEK(207)+PEEK(206)
40 SOURIS=256*PEEK(205)+PEEK(204)+2:DEF SEG=SEGMENT
50 KEY OFF:SCREEN 2:CLS
60 PRINT TAB(15)"MENU DEROULANT NO 1"TAB(48)"MENU DEROULANT NO 2"
70 GET(0,0)-(639,7),BANDE:CLS
80 LOCATE 1,7:PRINT"OPTION A":GET(0,0)-(167,7),OPTIONA
90 LOCATE 1,7:PRINT"OPTION B":GET(0,0)-(167,7),OPTIONB
100 LOCATE 1,7:PRINT"OPTION C":GET(0,0)-(167,7),OPTIONC
110 LOCATE 1,7:PRINT"OPTION D":GET(0,0)-(167,7),OPTIOND
120 CLS:BLOAD"MAISON.FIG":GET(0,0)-(639,7),HAUT
130 M1=0:CALL SOURIS(M1,M2,M3,M4)
140 M1=3:CALL SOURIS(M1,BOUTON,X,Y)
150 IF Y>=8 THEN PUT(0,0),HAUT,PSET:GOTO 140
160 M1=2:CALL SOURIS(M1,M2,M3,M4)
170 PUT(0,0),BANDE,PRESET
180 M1=1:CALL SOURIS(M1,M2,M3,M4)
190 IF BOUTON<>1 THEN 140
200 IF X>103 AND X<272 THEN GOSUB 240
210 IF X>=372 AND X<532 THEN GOSUB 490
220 GOTO 140
```

... et le sous-programme spécifique que voilà :

```
230 'Sous-programme du menu déroulant n°1------------------------------------
240 M1=2:CALL SOURIS(M1,M2,M3,M4)
250 GET(104,8)-(271,90),IMAGE
260 PUT(104,8),OPTIONA,PRESET
270 PUT(104,16),OPTIONB,PRESET
280 PUT(104,24),OPTIONC,PRESET
290 PUT(104,32),OPTIOND,PRESET
300 M1=1:CALL SOURIS(M1,M2,M3,M4)
310 M1=3:CALL SOURIS(M1,BOUTON,X,Y)
320 IF X<104 OR X>271 THEN 400
330 M1=2:CALL SOURIS(M1,M2,M3,M4)
340 IF Y<8 OR Y>=40 THEN PUT(104,8),OPTIONA,PRESET:PUT(104,16),OPTIONB,PRESET:
    PUT(104,24),OPTIONC,PRESET:PUT(104,32),OPTIOND,PRESET
350 IF Y>=8 AND Y<16 THEN PUT(104,8),OPTIONA,PSET:PUT(104,16),OPTIONB,PRESET:
    PUT(104,24),OPTIONC,PRESET:PUT(104,32),OPTIOND,PRESET
360 IF Y>=16 AND Y<24 THEN PUT(104,8),OPTIONA,PRESET:PUT(104,16),OPTIONB,PSET:
    PUT(104,24),OPTIONC,PRESET:PUT(104,32),OPTIOND,PRESET
370 IF Y>=24 AND Y<32 THEN PUT(104,8),OPTIONA,PRESET:PUT(104,16),OPTIONB,PRESET:
    PUT(104,24),OPTIONC,PSET:PUT(104,32),OPTIOND,PRESET
380 IF Y>=32 AND Y<40 THEN PUT(104,8),OPTIONA,PRESET:PUT(104,16),OPTIONB,PRESET:
    PUT(104,24),OPTIONC,PRESET:PUT(104,32),OPTIOND,PSET
390 M1=1:CALL SOURIS(M1,M2,M3,M4)
400 IF BOUTON<>1 THEN GOTO 310
410 CHOIX=0:IF X<104 OR X>271 OR Y<8 OR Y>=40 THEN 460
420 IF Y>=8 AND Y<16 THEN CHOIX=1
430 IF Y>=16 AND Y<24 THEN CHOIX=2
440 IF Y>=24 AND Y<36 THEN CHOIX=3
450 IF Y>=32 AND Y<40 THEN CHOIX=4
460 M1=2:CALL SOURIS(M1,M2,M3,M4)
470 PUT(104,8),IMAGE,PSET
480 M1=1:CALL SOURIS(M1,M2,M3,M4)
490 RETURN
```

Comment tout cela fonctionne ?... En ce qui concerne le programme principal, vous y reconnaissez la méthode d'affichage en vidéo inversée décrite précédemment. Dans ce cas précis, il s'agit d'une BANDE de texte situé dans le haut de l'écran (lignes 60 et 70) précisant qu'il existe deux menus déroulants différents, ainsi que les quatre OPTIONs A, B, C et D du premier des deux menus (lignes 90 à 120). Notez qu'en ce qui concerne ces OPTIONs, elles apparaissent de manière fugitive dans le coin supérieur gauche de l'écran, le temps d'être mise en tableau.

Ceci fait, la ligne 120 procède à l'effacement de l'écran et au chargement d'une image graphique(*) (c'est tout de même mieux qu'un écran vide...), dont la bande du HAUT est sauvegardée dans le tableau du même nom. Pourquoi cela ?... Mais afin de pouvoir reconstituer l'image dans son intégralité dès que la bande de texte disparaît (ligne 150). A ce sujet, il vous faut savoir que cette bande de texte, invisible la majorité du temps, devient visible dès que le curseur pénèbre dans la BANDE de 8 pixels de haut située dans le HAUT de l'écran (lignes 160 à 180). Notez également que le réaffichage de la portion du HAUT de l'image s'effectue avec l'option PSET de l'instruction PUT qui a pour effet d'afficher en vidéo normale le contenu du tableau, en écrasant au passage le fond d'écran existant, c'est-à-dire en l'occurence la BANDE de texte.

Côté affichage, seul celui du BOUTON de gauche est pris en compte par les lignes 190 à 210, la ligne 200 assurant le branchement vers le sous-programme du 1er menu déroulant et la ligne 210 vers le sous-programme du second (qu'il vous reste à écrire...)

(*) Cette image est extraite de la bibliothèque d'images livrées avec le logiciel de dessin PC PAINT.

Voyons un peu ce sous-programme de déroulement de menu !... Notez qu'il débute en sauvegardant la portion d'IMAGE (ligne 250) qui va être détruite par l'affichage des OPTIONs A à D de ce premier menu (lignes 260 à 290). Ceci fait, on procède à l'aide des lignes 340 à 380 à l'affichage sur fond noir de l'option sur laquelle passe le curseur.

Il ne vous reste donc plus qu'à cliquer sur l'option de votre choix et ce sont les lignes 400 à 450 qui se chargent alors d'affecter une valeur à la variable CHOIX. Ce dernier étant fait, il n'y a plus lieu de maintenir le menu déroulé et c'est pourquoi, après avoir effacé le curseur (ligne 460), on rétablit la portion d'IMAGE située sous le menu (ligne 470), ce qui a pour effet d'effacer le menu et, après avoir rétablit le curseur, on expédie le tout vers le programme principal.

La mise en œuvre de menus déroulants nécessite que soient sauvegardées dans des tableaux graphiques les portions d'images situées sous les options des menus.

DÉPLACEMENT D'ICÔNES SUR L'ÉCRAN

Et maintenant que le plus dur est fait, que diriez-vous si l'on se payait un petit déplacement d'icônes sur l'écran, histoire de se délasser un peu les méninges ?... Vous êtes pour ?... Alors, dans ces conditions, je vous propose de transformer votre IBM PC en synthétiseur de musique (en imagination, s'entend...) et à procéder aux différents réglages que celui-ci nécessite en vous servant de la souris comme d'un doigt pour pousser ou tirer le curseur des quatre potentiomètres que voici :

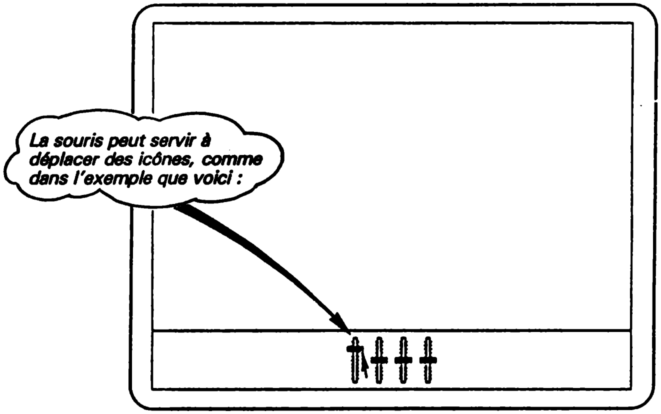


Fig. 10.9 : Réglage des potentiomètres avec une souris

Vous allez voir à quel point c'est pratique (et drôle...) à utiliser !... Tapez donc en vitesse (mais sans vous tromper...) le court programme que voici :

```
10 DEFINT A-Z:DIM POTENTIOMETRE(6)
20 POTENTIOMETRE$="C1R7BR2R3BR2R7DL5BL2L0BL2L3BL2L0BL2L5DR7BR2R3BR2R7"
30 DEF SEG=0:SEGMENT=256*PEEK(207)+PEEK(206)
40 SOURIS=256*PEEK(205)+PEEK(204)+2:DEF SEG=SEGMENT
50 KEY OFF:SCREEN 2:CLS
60 X1=262:Y1=182:X2=294:Y2=182:X3=326:Y3=182:X4=358:Y4=182
70 PRESET(X1,Y1):DRAW POTENTIOMETRE$
80 GET(X1,Y1)-(X1+21,Y1+2),POTENTIOMETRE
90 PUT(X1,Y1),POTENTIOMETRE
100 LINE(0,0)-(639,199),,B:LINE(0,167)-(639,167)
110 FOR X=X1 TO X4 STEP 32:LINE(X+8,171)-(X+13,195),,B:
    FOR Y=173 TO 193 STEP 2:PSET(X+6,Y):PSET(X+15,Y):NEXT:
    PUT(X,Y1),POTENTIOMETRE:NEXT
120 M1=0:CALL SOURIS(M1,M2,M3,M4)
130 M1=3:CALL SOURIS(M1,BOUTON,X,Y):IF BOUTON<>1 THEN 130
140 IF X>=X1 AND X<=X1+21 AND Y>=Y1-4 AND Y<=Y1+4 THEN GOSUB 170
150 GOTO 130
```

... suivi du encore plus court sous-programme de réglage du potentiomètre n° 1 que voilà :

```
160 'Sous-programme de réglage du potentiomètre n°1-------------------------
170 IF Y=Y1 THEN RETURN
180 M1=2:CALL SOURIS(M1,M2,M3,M4)
190 PUT(X1,Y1),POTENTIOMETRE
200 IF Y>Y1 AND Y1<192 THEN Y1=Y1+2
210 IF Y<Y1 AND Y1>172 THEN Y1=Y1-2
220 PUT(X1,Y1),POTENTIOMETRE
230 M1=1:CALL SOURIS(M1,M2,M3,M4)
240 REGLAGE=(192-Y1)/20
250 RETURN
```

Du point de vue fonctionnement, que trouve-t-on ?... Encore une fois, rien de bien particulier !... Après affectation de valeurs aux variables X1, Y1 à X4, Y4 de positionnement des quatre potentiomètres (ligne 60), vous trouvez les habituelles lignes de mise en tableau du curseur des **POTENTIOMETREs** (lignes 70 à 90), suivi du dessin de quatre d'entre-eux dans le bas de l'écran (ligne 110).

Ceci fait, vous rencontrez les habituelles lignes d'activation de la souris (ligne 120), de détection de sa position et de l'état de son **BOUTON** de gauche (ligne 130). Ce n'est que lorsque celui-ci est enfoncé que vous pouvez (enfin !...) accéder à la ligne 140 qui se charge de vérifier si le curseur pointe sur le potentiomètre n° 1 (pour les autres, il vous suffit d'écrire les trois lignes manquantes à la suite de celle-ci...). Si c'est le cas, vous avez le droit d'aller faire un aller et retour dans le sous-programme 170.

Or, que fait ce sous-programme ?... Il commence par vérifier (sait-on jamais...) si le curseur du potentiomètre ne serait pas déjà là où l'on voudrait qu'il soit (ligne 270). Si ce n'est pas le cas, on procède alors à l'extinction du curseur de la souris (ligne 180),

suivi à son tour par l'effacement de celui du POTENTIOMETRE (ligne 190). Ceci fait, on calcule à l'aide des lignes 200 et 210 si son déplacement doit s'effectuer vers le haut ou le bas, avant de le redessiner (ligne 220) et de rallumer celui de la souris (ligne 230). Quant à la ligne 80, elle se charge de transformer la position du curseur en une valeur de RÉGLAGE comprise entre 0 et 1.

Et voilà le travail !... A vous maintenant de faire mieux à partir des bases jetées dans ce chapitre !

Le déplacement d'icônes sur l'écran s'obtient en détectant lors du cliquetage de la souris si l'icône est située dans son périmètre d'attraction.

Solution des exercices sur le chapitre 10

1. Non, absolument pas avant chaque exécution d'un programme graphique !... Il suffit qu'il soit chargé, un point c'est tout !
2. Dans n'importe quel ordre du moment que le fichier BASICA est chargé en dernier.
3. A localiser le fichier MOUSE dans la mémoire, ce qui permet de déterminer le début du logiciel de commande de la souris.
4. La fonction 0 si la souris n'avait pas encore été activée ou la fonction 1 si le curseur a été éteint avec la fonction 2.
5. A l'enfoncement simultané des boutons-poussoirs.
6. 1 mickey/8 pixels.
7. Dans aucun des deux !... La position relative du point de référence du curseur de la souris se définit avec les paramètres d'entrée M2 et M3.

Exercices sur le chapitre 10

1. Est-il nécessaire de charger le fichier MOUSE avant chaque exécution d'un programme utilisant la souris :

☐ Non, absolument pas
☐ Oui, sans quoi le fichier MOUSE est inrepérable dans la mémoire.

2. Dans quel ordre faut-il charger le fichier MOUSE lorsque l'on procède à son autochargement avec AUTOEXEC. BAT ?

☐ Avant le fichier BASICA.
☐ Dans n'importe quel ordre.

3. A quoi servent les PEEK (204) à (207) ?

☐ A localiser le fichier MOUSE dans le mémoire
☐ A déterminer le début du logiciel de commande de la souris.

4. Quelle est la fonction qui rend visibles sur l'écran les déplacements de la souris ?

☐ La fonction 0
☐ La fonction 1
☐ La fonction 3

5. A quoi correspond la valeur 3 de l'attribut M2 de la fonction 3 ?

☐ A la position horizontale du curseur sur la 2e colonne
☐ A l'enfoncement simultané des boutons poussoirs

6. Quelle valeur confère à la souris une vitesse horizontale ultra rapide ?

☐ 1 mickey/8 pixels
☐ 32000 mickeys/8 pixels.

7. Où définit-on la position relative du point de référence du curseur de la souris ?

☐ Dans le masque de fond d'écran.
☐ Dans le masque du curseur

Réponses page 191

CHAPITRE XI

UNE APPLICATION GRAPHIQUE ENTRE MILLE :

LE LOGICIEL GRAFELEC DE DESSIN DE SCHÉMAS D'ÉLECTRONIQUE

Fort des connaissances acquises et des logiciels et sous-programmes utilitaires dont vous disposez dorénavant, vous voici à même de procéder à la phase ultime de votre démarche, à savoir la réalisation d'un logiciel graphique répondant à une application bien spécifique. Or, les applications qui mettent en jeu des graphiques sur ordinateur sont aussi innombrables et diversifiées que les cerveaux qui les conçoivent. C'est pourquoi à travers l'exemple du logiciel de la mise au propre de schémas d'électronique, le présent chapitre n'a d'autre but que de vous montrer comment assembler les éléments de base de cet ouvrage pour aboutir au résultat recherché.

Menu du chapitre 11

Inventaire des ressources graphiques.
Analyse d'une application.
Conception du menu du logiciel GRAFELEC.
Elaboration d'une grille de dessin.
Constitution de la bibliothèque de symboles
Sélection et mise en place des symboles
Rotations et sorties d'écran
Le problème des points de connexion
Le logiciel GRAFELEC

Quel est votre choix : ■

INVENTAIRE DES RESSOURCES GRAPHIQUES

Avant d'aborder la phase de conception du logiciel GRAFELEC de dessin de schémas d'électronique, il serait peut être bon de faire le point sur les outils de base dont nous disposont, ne serait-ce que pour nous rafraîchir un peu la mémoire (dix chapitres, c'est long...).

Il s'agit, par ordre d'entrée en scène (et non par ordre d'importance...) :

1°) *Des chaînes de caractères* destinées à produire, en association avec l'instruction **PAINT**, toute une gamme de densités de points de couleurs différentes servant au coloriage des figures géométriques (chapitre 3).

2°) *Du sous-programme INPUT* qui asure un contrôle rigoureux de la longueur des entrées, auquel vient s'ajouter l'affichage par défaut des chaînes existantes ainsi que diverses commandes comme par exemple le retour au menu principal avec la touche [ESC] (chapitre 4).

3°) *Du logiciel DATAGRAF* de représentation graphique des données (chapire 5).

4°) *Du gestionnaire FUSFORM* spécialement conçu pour créer et modifier le contenu des bibliothèques de formes graphiques (chapitre 6).

5°) *Du sous-programme PRINT* qui, associé à la *police de caractères POLICE.128* et à l'utilitaire de création de la dite police, permet l'affichage en page graphique à moyenne résolution des caractères en mode 80 colonnes (chapitre 7).

6°) *Du déplacement sous le contrôle du clavier* des formes graphiques d'un point à l'autre de l'écran et ce, sans altération du décor de fond (chapitre 8).

7°) *Du chargement de la sauvegarde et de l'impression* sur papier en vidéo normale ou inversé des images élaborées à l'écran (chapitre 9).

8°) *Des fenêtres graphiques, menus déroulants* et autres incrustations qu'il est possible de superposer à une image lorsque le besoin s'en fait sentir (chapitres 9 et 10)

9°) *De la souris* et de son maniement dans les logiciels de type graphique (chapitre 10).

De temps à autre, il est nécessaire de procéder à l'inventaire des outils graphiques dont on dispose.

ANALYSE D'UNE APPLICATION

Et nous voici arrivés au pied du mur !... C'est à ce moment là que l'on se pose alors l'éternelle question : « Par quel bout vais-je commencer ? ». L'on serait fortement tenté de répondre « Mais par le début ! », si cette palissade ne cachait pas une profonde réflexion sur la finalité de notre démarche.

En effet, le début en question, c'est de savoir si ce que l'on cherche à obtenir est réalisable avec les outils dont on dispose. Or, que cherche-t-on à obtenir dans le cas précis du logiciel **GRAFELEC** ?... Un moyen rapide et efficace de dessin de schémas d'électronique, d'abord sur écran, puis ensuite sur papier, nous dispensant ainsi du traditionnel dessin sur calque, avec té, rotrings, équerres et tout le « saint-frusquin ».

Or, que nous faut-il pour mettre en œuvre cette application purement graphique ?...

En tout premier lieu, il nous faut un menu qui, s'incrustant sur l'écran graphique pardessus le schéma électronique, assure le passage d'un mode de fonctionnement à un autre. Soyons assuré, car non seulement la chose est faisable, mais nous l'avons déjà réalisée à maintes reprises !...

Ensuite, il faut disposer d'une grille d'aide à la mise en place des symboles. Du bon choix du pas de cette grille dépendra toute la qualité des schémas que nous allons produire.

Après la grille, il nous faut une vaste bibliothèque de symboles électroniques. Pour cela, il nous suffit de faire appel aux formes graphiques. Non seulement, nous savons comment les créer et les utiliser mais de surcroît, nous disposont d'un gestionnaire de fichiers de formes pour nous aider à constituer la bibliothèque requise. Donc, pas de difficulté majeure de ce côté là !

Reste, toutefois à définir le procédé de sélection et de mise en place des symboles sur la grille. Il doit demeurer simple à mettre en œuvre tout en assurant une rapidité d'exécution des schémas bien supérieure à celle qu'il est possible d'obtenir avec les procédés traditionnels. Quelle solution sera retenue ?... Dans ce domaine, seule l'expérimentation nous fournira la réponse !

Parlant de mise en place des symboles, il nous faudra trouver un procédé astucieux pour les faire tourner de 90 degrés en 90 degrés. Utiliserons-nous le sous-langage DRAW avec toutes les limitations qu'il engendre (impossibilité de dessiner dans la couleur complémentaire du fond d'où perte de celui-ci, arrêt brutal du programme dès que l'on tente de dessiner une forme en dehors de l'écran, etc...) ou bien allons-nous inventer quelque procédé nouveau ?... Cela reste à définir !

Toujours dans le domaine des hypothèses, il nous faudra trouver une solution au sujet des points d'intersection des traits des schémas. Allons-nous devoir les dessiner un par un ou bien au contraire n'est-il pas possible de mettre au point une méthode de tracé automatique ?... Sur ce point précis, c'est plus la réflexion que l'expérimentation qui nous fournira la réponse !

Avant de se lancer à corps perdu dans la frappe d'un programme, il y a lieu de procéder à une analyse aussi complète que possible de l'application envisagée.

CONCEPTION DU MENU DU LOGICIEL GRAFELEC

Comme nous envisageons d'élaborer une image qui ne sera rien d'autre qu'un schéma électronique, il semble que la meilleure solution soit de concevoir un menu principal de type « fenêtre » qui viendra s'incruster sur l'image quand on aura besoin de lui et qui disparaîtra dès son office rempli. On évitera ainsi de devoir passer sans cesse de l'écran texte à l'écran graphique chaque fois que l'on désire changer d'option.

Maintenant que ce premier choix est fait, il reste à déterminer le nombre d'options différentes que devra pouvoir vous présenter ce menu-fenêtre. Après mure réflexion, il semble qu'il y en ait cinq :

* Le **DESSIN** du schéma électronique.

* L'effacement global de l'écran (et donc du schéma) lorsque l'on se rend compte qu'après s'être trompé il est plus rapide de tout effacer d'un seul coup avec la **GOMME** que symbole par symbole.

* Le **CHARGE**ment d'un schéma électronique sur l'écran, que ce soit pour le terminer, y apporter quelques modifications ou plus simplement parce qu'il va servir de base à l'élaboration d'un schéma plus complexe.

* La **SAUVEGARDE** d'un schéma électronique sur disquette, histoire de ne pas perdre le fruit de son travail.

* La sortie du programme quand on estime que c'est la **FIN**.

Remarque : *L'existence de la fonction de recopie d'écran intégrée à l'appareil fait qu'il est inutile de prévoir l'impression des schémas dans le menu. il suffira d'appuyer le moment venu sur* [⇧] – [PrtSc].

Tout ceci laisse entrevoir la possibilité de travailler en couleurs, c'est-à-dire en moyenne résolution, le nombre total de lettres des options du menu ne dépassant pas 30. Dans ces conditions, nous allons pouvoir simuler (enfin !...) le menu-fenêtre avec les quelques lignes que voici :

```
10 DIM FENETRE(660),MENU(160),DESSIN(28),GOMME(25)
20 DIM CHARGE(28),SAUVEGARDE(42),FIN(18)
30 KEY OFF:SCREEN 1:COLOR 8,1:CLS
40 CHOIX=1
50 'Menu principal--------------------------------------------------------
60 GET(0,167)-(319,199),FENETRE
70 LINE(0,167)-(319,199),3,B:LINE(1,168)-(318,198),0,BF
80 LOCATE 22,2:PRINT"DESSIN  GOMME  CHARGE  SAUVEGARDE  FIN"
90 GET(1,168)-(318,175),MENU
100 GET(0,168)-(63,174),DESSIN
110 GET(64,168)-(119,174),GOMME
120 GET(120,168)-(183,174),CHARGE
130 GET(184,168)-(279,174),SAUVEGARDE
140 GET(280,168)-(319,174),FIN
150 PUT(1,168),MENU,PRESET
160 IF CHOIX=1 THEN PUT(0,168),DESSIN,PSET
170 IF CHOIX=2 THEN PUT(64,168),GOMME,PSET
180 IF CHOIX=3 THEN PUT(120,168),CHARGE,PSET
190 IF CHOIX=4 THEN PUT(184,168),SAUVEGARDE,PSET
200 IF CHOIX=5 THEN PUT(280,168),FIN,PSET
210 TOUCHE$=INKEY$:IF TOUCHE$="" THEN 210
220 ASCII=ASC(TOUCHE$):IF ASCII=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1))
230 IF ASCII=75 THEN CHOIX=CHOIX-1:IF CHOIX<1 THEN CHOIX=5
240 IF ASCII=77 THEN CHOIX=CHOIX+1:IF CHOIX>5 THEN CHOIX=1
250 IF ASCII<>13 THEN 150
260 ON CHOIX GOTO 1000,2000,3000,4000
270 CLS:KEY ON:END
```

Quant au résultat qu'elles procurent, le voilà :

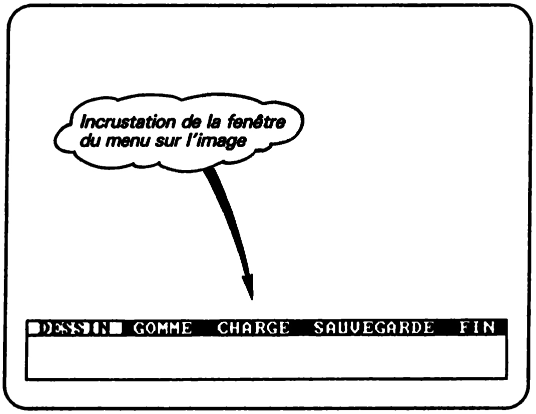


Fig. 11.1 : Création d'un menu de type « fenêtre »

Comme vous pouvez le constater par vous-même, ces quelques lignes ne font que reprendre les notions développées au cours des chapitres 9 et 10. Après une sauvegarde en tableau graphique de la portion d'image située sous la FENÊTRE (ligne 60), le programme dessine celle-ci (ligne 70), ainsi que la liste des options du MENU (ligne 80).

A quoi servent les lignes 90 à 140 ?... Mais à sauvegarder les options dans les tableaux graphiques, soit en totalité comme à la ligne 90, soit individuellement (lignes 100 à 140) de façon à pouvoir le dessiner en mode normal ou inverse lorsque le besoin s'en fera sentir.

Or, point n'est besoin d'attendre longtemps pour cela puisque après avoir détecté l'enfoncement des touches [←], [→] ou [ENTER] (lignes 210 et 220), le programme redessine à la ligne 150 la totalité des options en mode inverse (cela correspond à l'effacement du CHOIX précédent) avant de procéder au dessin en vidéo normale du nouveau CHOIX effectué au clavier (lignes 160 à 200 et 230 à 250).

Dans le cas du logiciel GRAFELEC, un menu de type « fenêtre » semble être le mieux adapté à l'application envisagée.

ÉLABORATION D'UNE GRILLE DE DESSIN

Quand nous parlons d'élaborer une grille d'aide au dessin, nous sous-entendons qu'il s'agit d'une grille de points, c'est-à-dire d'une grille assez discrète pour ne pas géner le dessinateur tout en étant suffisamment visible pour être efficace. En aucun cas, une grille de traits continus ou pointillés ne conviendrait !

Ce point étant acquis, il reste à définir le pas de cette grille de points. Trop grand, il autorisera la création de symboles relativement précis mais de taille telle que l'on ne pourra en disposer qu'une faible quantité sur l'écran. Trop petit, par contre, il permettra d'en implanter un plus grand nombre au détriment de la qualité de représentation des symboles.

Il semble donc que la solution idéale réside dans le compromis taille/précision des symboles. Pour nous aider dans notre recherche, je suggère que l'on s'oriente vers la détermination d'un pas qui soit à la fois un sous-multiple du nombre de pixels horizontaux et verticaux de l'écran. Un rapide calcul montre que :

320 pixels = 8.8.5
200 pixels = 8.5.5.

Dans ces conditions, la solution est vite trouvée !... Le pas est à choisir parmi les valeurs 5, 8, 10, 20 et 40. La valeur 5 étant trop faible et les valeurs 20 et 40 trop grandes, il reste à choisir entre un pas de 8 ou de 10 pixels. A mon avis, la valeur de 8 devrait convenir (j'ai un faible pour les nombres d'or binaires...).

Reste donc à simuler cette grille de dessin. Là, il faut avouer que ce n'est pas trop fatiguant, la preuve :

```
10 KEY OFF:SCREEN 1:COLOR 8,1:CLS
20 FOR J=8 TO 192 STEP 8
30 FOR I=8 TO 312 STEP 8
40 PSET(I,J),2
50 NEXT I,J
```

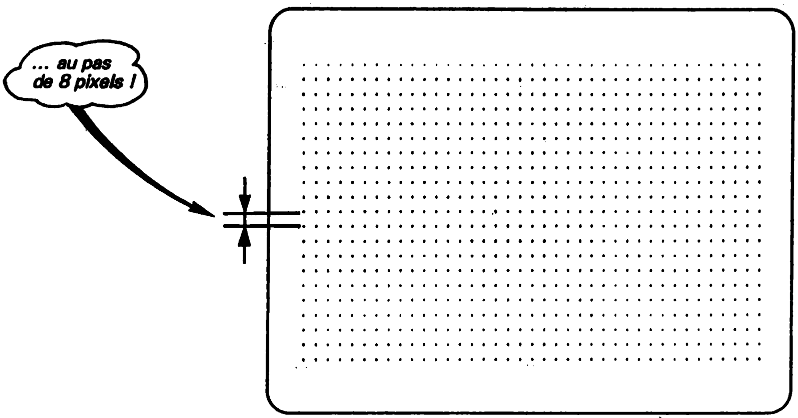


Fig. 11.2 : Grille de dessin au pas de 8

Au fait, la couleur violette des points vous plait-elle ?... Je le souhaite car c'est la seule qui m'a paru répondre à l'aspect de discrétion souhaitée. En conséquence, nous dessinerons les schémas électroniques en blanc (c'est la seule couleur qui s'imprime correctement sur le papier...).

CONSTITUTION DE LA BIBLIOTHÈQUE DE SYMBOLES

La constitution d'une bibliothèque de symboles correspondant aux composants électroniques ne présente guère plus de difficultés que celle de la bibliothèque de formes graphiques que nous avons dû réaliser au chapitre 7 pour permettre l'écriture des lettres verticales en page graphique. Cela ne réclame qu'un peu d'huile de coude !

Comme celle-ci ne nous manque pas, il ne nous reste plus qu'à relever les manches et à nous procurer du papier quadrillé. Pourquoi du papier quadrillé ?... Parce qu'il va falloir que nous dessinions chaque symbole pixel par pixel avant de pouvoir passer à l'opération de codage.

Les premiers symboles à passer sur la sellette sont ceux des trois composants passifs fondamentaux que sont la résistance, le condensateur et l'inductance. Vous noterez qu'en ce qui concerne la résistance, il va falloir prévoir un curseur mobile de façon à pouvoir la transformer en un potentiomètre lorsque le besoin s'en fera sentir (voir figure 11.3).

Côté symbole de condensateur, il va falloir en dessiner deux modèles différents selon que ceux-ci sont du type non polarisé ou électrochimique. Egalement, il y a lieu de prévoir pour l'inductance la possibilité de disposer d'un point milieu ainsi que d'un noyau magnétique, ne serait-ce que pour représenter les transformateurs d'alimentation.

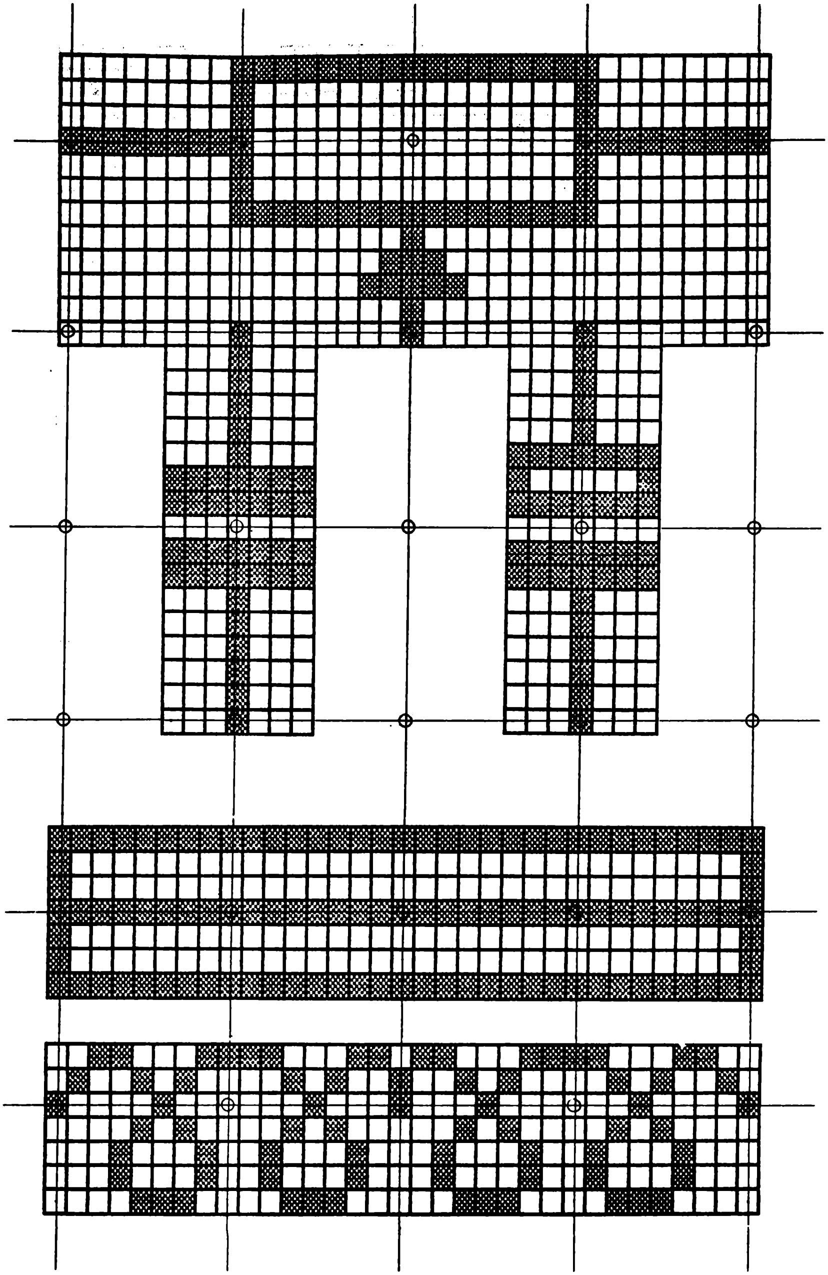


Fig. 11.3 : Symboles de quelques composants passifs élaborés par rapport à la grille de points au pas de 8.

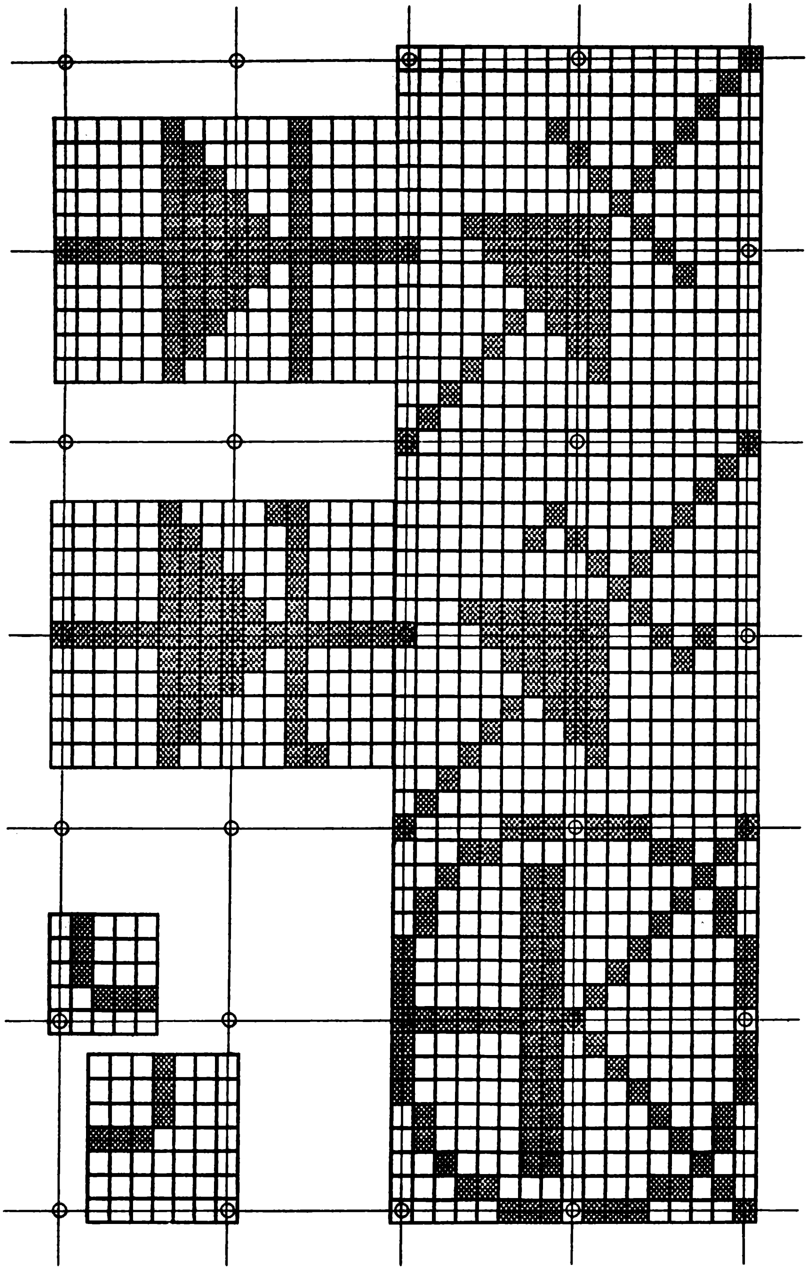


Fig. 11.4 : Symboles de quelques composants actifs élaborés par rapport à la grille de points au pas de 8.

Après les symboles des composants passifs, il est temps de passer à ceux des composants actifs. Toutefois, pour éviter de transformer ce chapitre en un cours d'électronique, seuls quelques-uns d'entre-eux seront présentés. Pour les autres(*), adressez-vous à votre fournisseur habituel (c'est-à-dire à vous-même...).

En ce qui concerne la diode (ou redresseur), il va falloir en prévoir deux versions selon qu'elle est employée seule ou dans un pont de redressement (voir figure 11.4). Il en est de même pour la diode zener pour qui il faut également prévoir deux versions différentes. Quant au transistor, mieux vaut ne pas spécifier dès le départ s'il est du type npn ou pnp. Il suffira d'utiliser deux flèches différentes pour cela.

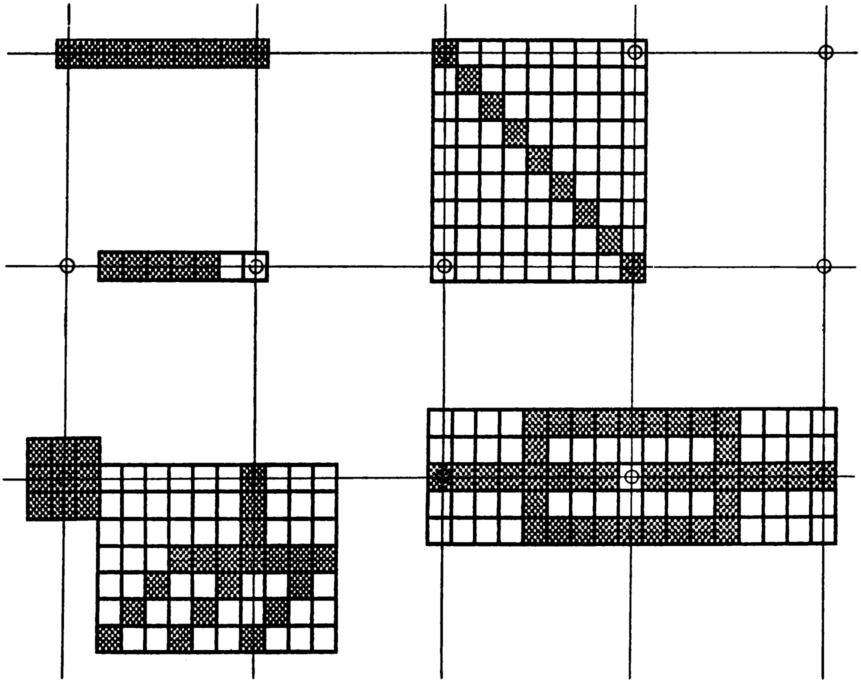


Fig. 11.5 : Quelques symboles électriques indispensables.

Enfin, après les composants passifs et actifs, il reste à voir les symboles électriques tels que ceux du fil de liaison, du point de soudure, de la masse du montage, du fusible de protection, etc, etc... (voir figure 11.5).

(*) Le logiciel GRAFELEC a été spécialement étudié par l'auteur pour le dessin des schémas électroniques de l'ouvrage « Calcul pratique des alimentations ». Ceci explique pourquoi le nombre de symboles est limité à l'essentiel.

Fort de tout ceci, il ne reste plus qu'à constituer la bibliothèque de symboles. Pour cela, il faut mettre au point un utilitaire de constitution de fichier de formes graphiques tel que celui décrit au chapitre 6. Quant au nom du fichier, pourquoi ne pas l'appeler **SYMBOLES. LIB** ?

```
10 DIM SYMBOLE$(19)
20 SYMBOLE$(1)="C3URD2L2U2":'Connexion
30 SYMBOLE$(2)="C3R8":'Trait
40 SYMBOLE$(3)="C3BR2R4":'Pointillé
50 SYMBOLE$(4)="C3D3L3R6G3BU3G3BU3G3":'Masse
60 SYMBOLE$(5)="C3R8U3R16D3R8L8D3L16U3":'Résistance
70 SYMBOLE$(6)="C3U4GR2FL4":'Curseur
80 SYMBOLE$(7)="C3D6L3UR6DL3BD2L3DR6UL3D7":'Condensateur non polarisé
90 SYMBOLE$(8)="C3D5L3D2R6U2L3BD4L3DR6UL3D7":'Chimique
100 SYMBOLE$(9)="C3E3R2F4DGL2HUE4R2F4DGL2HUE4R2F4DGL2HUE4R2F4DGL2HUE4R2F3":
                    'Inductance
110 SYMBOLE$(10)="C3BR4U2R24D2L24D2R24U2":'Noyau
120 SYMBOLE$(11)="C3R5D5U10FD8EU6FD4EU2FRD5U10D5R5":'Redresseur
130 SYMBOLE$(12)="C3F6E3G6RE5DG4RE3DG2REDFE3G6E3F6":'Pont
140 SYMBOLE$(13)="C3R5U5D10EU8FD6EU4FD2EBU5RD10RBU5R4":'Zener
150 SYMBOLE$(14)="C3F6E3G6RE5DG4RE3DG2REDBE3FG6FBE3F5":'Diode
160 SYMBOLE$(15)="C3U3EUE2RER6FRF2DFD6GDG2LGL6HLH2UHU3R6U6RD12LU6R2E8G8F8":
                     'Transistor
170 SYMBOLE$(16)="C3BFBR3L2GD2":'pnp
180 SYMBOLE$(17)="C3BF5BU3D2GL2":'npn
190 SYMBOLE$(18)="C3R16D16L16U16":'Régulateur
200 SYMBOLE$(19)="C3R4U2R8D4L8U2R12":'Fusible
210 FICHIER$="SYMBOLES.LIB":OPEN FICHIER$ FOR OUTPUT AS#1
220 FOR I=1 TO 19:WRITE#1,SYMBOLE$(I):NEXT:CLOSE#1
```

Le fichier **SYMBOLES. LIB** étant constitué (un petit coup de périscope au répertoire pour s'en assurer...), il n'y a plus qu'à faire quelques essais de dessin des symboles à l'écran. C'est ainsi que l'on découvre (avec horreur...) que l'on avait oublié que la machine ne respecte pas les proportions des formes qu'elle dessine (rappelez-vous le facteur de correction de 5/6...).

Ce défaut étant surtout visible sur les redresseurs et les diodes zener en pont (symboles 12 et 14) et ce, du fait des diagonales à 45 degrés qui ne le sont plus, nous confirmons notre intention de vouloir travailler en mode graphique à moyenne résolution. Pourquoi cela ?... Parce que la déformation qui s'opère lors de l'impression sur papier de l'écran à moyenne résolution compense (en partie seulement...) le défaut de proportionnalité de l'affichage sur cet écran. Ce n'est pas génial, mais c'est le mieux que l'on puisse faire !

Le regroupement en une bibliothèque des nombreux symboles que nécessite le logiciel GRAFELEC évite d'encombrer inutilement le corps du programme principal.

SÉLECTION ET MISE EN PLACE RAPIDE DES SYMBOLES

Ce point étant acquis, il nous faut maintenant déterminer la méthode de sélection et de mise en place des symboles.

Allons-nous chercher à utiliser la souris pour mettre en place un symbole sur l'écran après avoir été le chercher dans une vitrine ?... Certainement pas !... Si le déplacement d'icônes avec une souris est toujours spectaculaire à observer, il n'est absolument pas évident que cela permette une mise en place rapide des symboles sur la grille de points.

Une méthode plus efficace (sur le plan rapidité s'entend...) doit pouvoir être élaborée à partir des touches fléchées et des touches ⊞ et ⊟ servant à sélectionner le symbole à dessiner.

Petite difficulté à l'horizon !... Etant donné qu'il va falloir déplacer des symboles par-dessus la grille de points, ainsi que par-dessus d'autres symboles déjà mis en place, il devient impératif qu'ils soient dessinés dans la couleur complémentaire du fond. Pour cela, nous allons devoir faire appel aux tableaux graphiques comme cela vous a été montré dans le chapitre 9.

Seconde petite difficulté !... Les tableaux graphiques ne peuvent être dimensionnés qu'à une seule dimension. Cela exclut donc la possibilité d'appeler chaque symbole à l'aide d'un indice différent. Il va falloir faire appel à un véritable sous-programme chargé d'associer chaque tableau graphique au **NUMERO** du symbole qui lui correspond.

Voyons un peu tout cela en détails !... Limité à 4 symboles, le programme de simulation ressemble à ceci :

```
10 NOMBRE=19:DIM SYMBOLE$(NOMBRE)
20 DIM CONNEXION(1),TRAIT(2),POINTILLE(1),MASSE(6)
30 FICHIER$="SYMBOLES.LIB":OPEN FICHIER$ FOR INPUT AS#1
40 FOR I=1 TO 19:INPUT#1,SYMBOLE$(I):NEXT:CLOSE#1
50 X=8:Y=8:NUMERO=1
60 KEY OFF:SCREEN 1:COLOR 8,1:CLS
70 PRESET(X,Y):DRAW SYMBOLE$(1):GET(X-1,Y-1)-(X+1,Y+1),CONNEXION:CLS
80 PRESET(X,Y):DRAW SYMBOLE$(2):GET(X,Y)-(X+8,Y),TRAIT:CLS
90 PRESET(X,Y):DRAW SYMBOLE$(3):GET(X+2,Y)-(X+6,Y),POINTILLE:CLS
100 PRESET(X,Y):DRAW SYMBOLE$(4):GET(X-6,Y)-(X+3,Y+6),MASSE:CLS
110 FOR J=8 TO 192 STEP 8:FOR I=8 TO 312 STEP 8:PSET(I,J),2:NEXT I,J
120 GOSUB 250
130 ASCII=0:TOUCHE$=INKEY$:IF TOUCHE$="" THEN 150
140 ASCII=ASC(TOUCHE$):IF ASCII=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1))
150 GOSUB 250
160 IF ASCII=32 THEN 130
170 IF ASCII=43 THEN NUMERO=NUMERO+1:IF NUMERO>NOMBRE THEN NUMERO=1
180 IF ASCII=45 THEN NUMERO=NUMERO-1:IF NUMERO<1 THEN NUMERO=NOMBRE
190 IF ASCII=72 AND Y>8 THEN Y=Y-8
200 IF ASCII=75 AND X>8 THEN X=X-8
210 IF ASCII=77 AND X<312 THEN X=X+8
220 IF ASCII=80 AND Y<192 THEN Y=Y+8
230 GOTO 120
240 'Sous-programme d'affichage des symboles--------------------------------
250 IF NUMERO=1 THEN PUT(X-1,Y-1),CONNEXION
260 IF NUMERO=2 THEN PUT(X,Y),TRAIT
270 IF NUMERO=3 THEN PUT(X+2,Y),POINTILLE
280 IF NUMERO=4 THEN PUT(X-6,Y),MASSE
290 RETURN
```

Rien de bien nouveau, n'est-ce pas ?... Vous avez reconnu le dimensionnement des tableaux graphiques (lignes 10 et 20), le chargement de la bibliothèque de symboles (lignes 30 et 40), leur transfert en tableaux personnalisés (lignes 70 à 100) et le dessin de la grille de points (ligne 110).

Côté sélection et mise en place des symboles, là aussi, rien de neuf !... Tout au plus pouvez-vous remarquer (on aurait du mal à ne pas le voir...) le sous-programme d'affichage des symboles (lignes 240 à 290) que l'on appelle à deux reprises, une première fois pour dessiner le symbole (ligne 120) et une seconde pour l'effacer (ligne 150).

Pour le reste, il s'agit de l'habituelle détection des touches enfoncées (lignes 130 et 140), suivie des lignes 160 à 220 de commande (sélection, déplacement et dessin d'un symbole). Notez qu'en ce qui concerne le dessin ou l'effacement d'un symbole, c'est la BARRE d'espacement qui, obligeant le programme à sauter directement à la ligne 130, fait que celui-ci saute une étape et dessine ce qui aurait du être effacé et vice-versa par sa ligne 150.

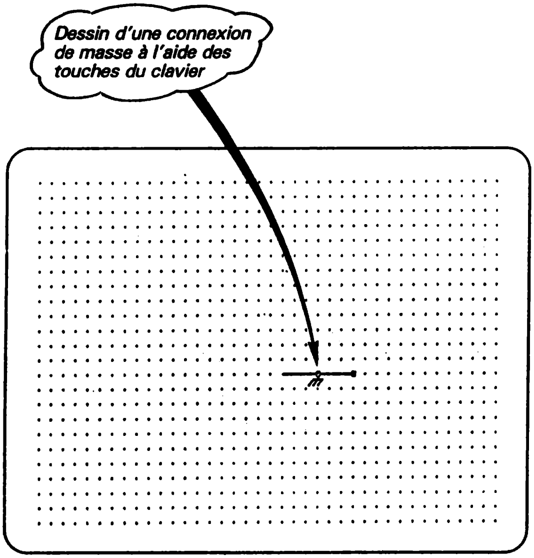


Fig. 11.6 : Sélection et mise en place des symboles

Afin de pouvoir être déplacés par-dessus le schéma électronique en cours d'élaboration, les symboles sont transférés dans des tableaux graphiques.

ROTATIONS ET SORTIES D'ÉCRAN

Du fait de l'emploi des tableaux graphiques à la place de l'instruction **DRAW** (incapable de dessiner dans la couleur complémentaire du fond...), nous perdons la possibilité de faire tourner les symboles de 90 en 90 degrés. Pour retrouver cette possibilité, il va falloir de nouveau faire appel à un sous-programme spécialement conçu pour cela.

Quelle méthode allons-nous employer ?...Toujours la même, c'est-à-dire celle des tableaux graphiques !... L'astuce (si tant est que cela en est une...) va consister à dessiner la forme dans ses 4 positions avec l'instruction **DRAW** et à mémoriser chaque position dans un tableau distinc. Bien entendu, certaines formes, de par leurs symétries, ne nécessiteront pas 4 tableaux distincts, mais se contenteront de 2 tableaux (traits, résistance, fusible, etc...), voire d'un seul (le point de connexion).

Mais, ce n'est pas tout !... Le dessin des tableaux graphiques avec l'instruction **PUT** ne tolère à aucun moment qu'un morceau quelconque du tableau puisse sortir de l'écran. C'est l'arrêt immédiat du programme qui en résulte. Or, cela peut se produire d'autant plus aisément que la longueur et la hauteur des symboles varie d'un symbole à l'autre selon son angle de rotation. Comment allons-nous procéder pour résoudre ce problème ?

Encore une fois, il va falloir créer un sous-programme spécialement chargé d'éviter les sorties d'écran des symboles. Une solution simple semble être de dissocier l'incrémentation des variables X et Y de la détection de franchissement des butées et d'opérer cette dernière dans un sous-programme adapté à cette fonction. Ainsi, il devient à tout moment possible de modifier la valeur des butées selon le symbole choisi et l'orientation qu'il peut avoir.

Afin de vérifier si cette théorie veut bien fonctionner, nous allons procéder à quelques essais sur les quatre premiers symboles de la bibliothèque :

```
10 NOMBRE=19:DIM SYMBOLE$(NOMBRE)
20 DIM CONNEXION(1),TRAITO(2),TRAIT3(3),POINTILLEO(1),POINTILLE3(2)
30 DIM MASSEO(6),MASSE1(5),MASSE2(6),MASSE3(5)
40 FICHIER$="SYMBOLES.LIB":OPEN FICHIER$ FOR INPUT AS#1
50 FOR I=1 TO 19:INPUT#1,SYMBOLE$(I):NEXT:CLOSE#1
60 X=8:Y=8:NUMERO=1
70 KEY OFF:SCREEN 1:COLOR 8,1:CLS
80 PRESET(X,Y):DRAW SYMBOLE$(1):GET(X-1,Y-1)-(X+1,Y+1),CONNEXION:CLS
90 PRESET(X,Y):DRAW SYMBOLE$(2):GET(X,Y)-(X+8,Y),TRAITO:CLS
100 PRESET(X,Y):DRAW "A3"+SYMBOLE$(2):GET(X,Y)-(X+3,Y+8),TRAIT3:CLS
110 PRESET(X,Y):DRAW SYMBOLE$(3):GET(X+2,Y)-(X+6,Y),POINTILLEO:CLS
120 PRESET(X,Y):DRAW "A3"+SYMBOLE$(3):GET(X,Y+2)-(X,Y+6),POINTILLE3:CLS
130 PRESET(X,Y):DRAW SYMBOLE$(4):GET(X-6,Y)-(X+3,Y+6),MASSEO:CLS
140 PRESET(X,Y):DRAW "A1"+SYMBOLE$(4):GET(X,Y-3)-(X+6,Y+6),MASSE1:CLS
150 PRESET(X,Y):DRAW "A2"+SYMBOLE$(4):GET(X-3,Y-6)-(X+6,Y),MASSE2:CLS
160 PRESET(X,Y):DRAW "A3"+SYMBOLE$(4):GET(X-6,Y-6)-(X,Y+3),MASSE3:CLS
170 FOR J=8 TO 192 STEP 8:FOR I=8 TO 312 STEP 8:PSET(I,J),2:NEXT I,J
180 GOSUB 330
190 ASCII=0:TOUCHE$=INKEY$:IF TOUCHE$="" THEN 210
200 ASCII=ASC(TOUCHE$):IF ASCII=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1))
210 GOSUB 330
220 IF ASCII=32 THEN 190
230 IF ASCII=82 THEN ROTATION=ROTATION+1:IF ROTATION>3 THEN ROTATION=0
240 IF ASCII=43 THEN NUMERO=NUMERO+1:IF NUMERO>NOMBRE THEN NUMERO=1
```

```
250 IF ASCII=45 THEN NUMERO=NUMERO-1:IF NUMERO<1 THEN NUMERO=NOMBRE
260 IF ASCII=72 AND Y>8 THEN Y=Y-8
270 IF ASCII=75 AND X>8 THEN X=X-8
280 IF ASCII=77 THEN X=X+8
290 IF ASCII=80 THEN Y=Y+8
300 GOSUB 440
310 GOTO 180
320 'Sous-programme d'affichage des symboles-----------------------------------
330 IF NUMERO=1 THEN PUT(X-1,Y-1),CONNEXION
340 IF NUMERO=2 AND (ROTATION=0 OR ROTATION=2) THEN PUT(X,Y),TRAITO
350 IF NUMERO=2 AND (ROTATION=1 OR ROTATION=3) THEN PUT(X,Y),TRAIT3
360 IF NUMERO=3 AND (ROTATION=0 OR ROTATION=2) THEN PUT(X+2,Y),POINTILLEO
370 IF NUMERO=3 AND (ROTATION=1 OR ROTATION=3) THEN PUT(X,Y+2),POINTILLE3
380 IF NUMERO=4 AND ROTATION=0 THEN PUT(X-6,Y),MASSEO
390 IF NUMERO=4 AND ROTATION=1 THEN PUT(X,Y-3),MASSE1
400 IF NUMERO=4 AND ROTATION=2 THEN PUT(X-3,Y-6),MASSE2
410 IF NUMERO=4 AND ROTATION=3 THEN PUT(X-6,Y-6),MASSE3
420 RETURN
430 'Sous-programme de traitement des sorties d'écran-------------------------
440 XMAX=312:YMAX=192
450 IF NUMERO=2 AND (ROTATION=0 OR ROTATION=2) THEN XMAX=304
460 IF NUMERO=2 AND (ROTATION=1 OR ROTATION=3) THEN YMAX=184
470 IF NUMERO=3 AND (ROTATION=0 OR ROTATION=2) THEN XMAX=304
480 IF NUMERO=3 AND (ROTATION=1 OR ROTATION=3) THEN YMAX=184
490 IF X>XMAX THEN X=XMAX
500 IF Y>YMAX THEN Y=YMAX
510 RETURN
```

Comme vous pouvez le constater, il a fallu introduire un certain nombre de modifications au programme précédent pour aboutir à nos fins. Ces modifications portent sur les lignes 90 à 160 de mise en tableau des symboles, les lignes 280 et 290 d'incrémentation des variables X et Y, les lignes 340 et 410 du sous-programme d'affichage des symboles et pour finir (par le début...), les lignes 20 et 30 de dimensionnement des tableaux. Bref, pratiquement toutes les lignes ont du être retouchées !

A cela, il faut rajouter la ligne 230 de **ROTATION** des symboles ainsi que la ligne d'appel du sous-programme de traitement des sorties d'écran (ligne 300). Notez au passage la position de cette ligne qui, placée à la suite de la dernière ligne de commande, permet de « rectifier le tir » avant que le programme ne tente de dessiner un symbole en dehors de l'écran.

Ceci dit, il est temps de vérifier si tout cela fonctionne correctement !... Horreur et abominations, la **ROTATION** des symboles entraîne une déformation de ceux-ci (c'est surtout visible sur le symbole de masse...) !

Qu'est-ce qui peut causer une telle déformation ?... Mais l'instruction **DRAW** du Basic Microsoft !... Alors, non seulement incapable de dessiner dans la couleur complémentaire du fond, voilà maintenant qu'elle déforme les formes graphiques dès qu'elle tente de les faire tourner !

Existe-t-il un remède à cet état de faits ?... Oui, bien sûr !... Il suffit de ne pas utiliser l'instruction **DRAW** pour faire tourner les symboles, ce qui bien entendu nous oblige à devoir coder les différentes positions qu'ils peuvent prendre (pourquoi faire simple quand on peut faire compliqué...)

Les insuffisances des instructions DRAW et PUT nécessitent l'emploi de sous-programmes spécifiques de rotation et de sortie d'écran des symboles.

LE PROBLÈME DES POINTS DE CONNEXION

Avant de passer à la frappe du programme, il nous reste encore un problème à résoudre, celui des points de connexion !... De quoi s'agit-il exactement ?... Jetez donc un coup d'œil à la modeste tentative de dessin d'une connexion de mase de la figure 11.6. Que constatez-vous ?... Que le point de connexion qui devrait être plein ne l'est pas !

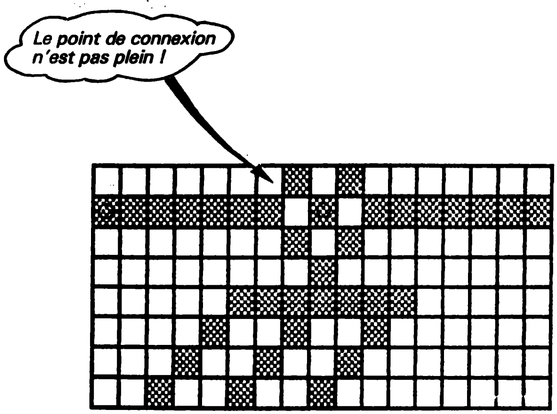


Fig. 11.7 : Dessin d'un point de connexion

Pourquoi en est-il ainsi ?... Parce qu'on le dessine dans la couleur complémentaire du fond et qu'en l'occurence, trois des pixels du fond correspondent respectivement aux amorces des deux traits horizontaux et de la prise de masse. De ce fait, ces trois pixels qui étaient allumés avant que l'on ne dessine le point ne le sont plus après qu'il ait été dessiné.

La solution ?... Ce pourrait être de ne pas dessiner le point dans la couleur complémentaire du fond, mais bien au contraire en écrasant tout ce qui peut se trouver en dessous. Pour mémoire, c'est l'option PSET de l'instruction PUT qui permet d'obtenir cet « écrasement ».

L'inconvénient qui en résulte ?... C'est qu'après avoir effacé le point (lors d'une modification de schéma, par exemple...), nous voyons apparaître à sa place un « trou noir » de 9 pixels. La question qui se pose alors est de savoir comment rétablir les pixels du fond qui ont été écrasés lors du dessin du point ?

La première idée qui me vient à l'esprit serait d'effectuer un test sur les pixels (il y en a 8 au total...) situés à la périphérie immédiate du trou noir. Ce faisant, on serait en mesure de déterminer l'emplacement des traits que le point reliait précédemment et par là-même, de reconstituer les pixels manquants du trou noir.

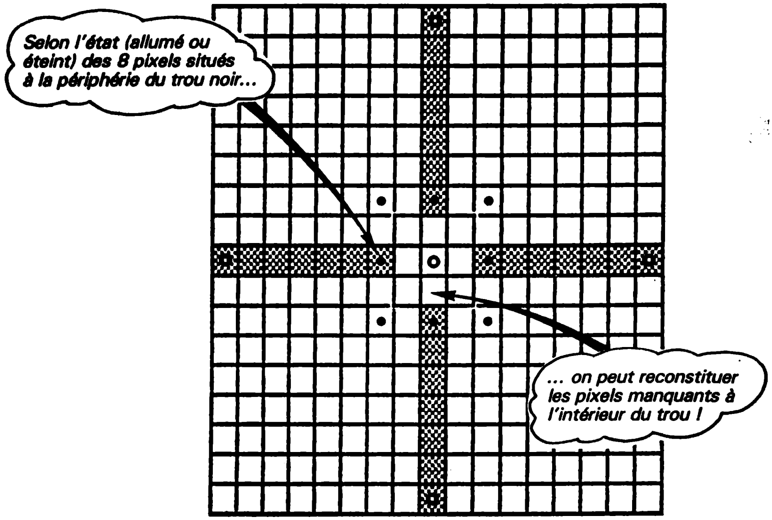


Fig. 11.8 : Reconstitution des pixels manquants

On peut même aller plus loin sur cette voie !... Plutôt que d'effectuer ce traitement sur les points de connexion un nombre incalculable de fois, pourquoi ne pas attendre l'effacement de la grille (qu'il faut réaliser avant la sauvegarde du schéma...) pour procéder à ce traitement. On y gagnerait à la fois en simplicité d'écriture du programme et en rapidité d'exécution !

Aussitôt dit, aussitôt fait !... Voici le résultat de nos cogitations :

```
10 NOMBRE=19:DIM SYMBOLE$(NOMBRE)
20 DIM CONNEXION(1),TRAITO(2),TRAIT3(3),POINTILLEO(1),POINTILLE3(2)
30 DIM MASSEO(6),MASSE1(5),MASSE2(6),MASSE3(5)
40 FICHIER$="SYMBOLES.LIB":OPEN FICHIER$ FOR INPUT AS#1
50 FOR I=1 TO 19:INPUT#1,SYMBOLE$(I):NEXT:CLOSE#1
60 X=8:Y=8:NUMERO=1
70 KEY OFF:SCREEN 1:COLOR 8,1:CLS
80 PRESET(X,Y):DRAW SYMBOLE$(1):GET(X-1,Y-1)-(X+1,Y+1),CONNEXION:CLS
90 PRESET(X,Y):DRAW SYMBOLE$(2):GET(X,Y)-(X+8,Y),TRAITO:CLS
100 PRESET(X,Y):DRAW "A3"+SYMBOLE$(2):GET(X,Y)-(X+3,Y+8),TRAIT3:CLS
110 PRESET(X,Y):DRAW SYMBOLE$(3):GET(X+2,Y)-(X+6,Y),POINTILLEO:CLS
120 PRESET(X,Y):DRAW "A3"+SYMBOLE$(3):GET(X,Y+2)-(X,Y+6),POINTILLE3:CLS
130 PRESET(X,Y):DRAW SYMBOLE$(4):GET(X-6,Y)-(X+3,Y+6),MASSEO:CLS
140 PRESET(X,Y):DRAW "A1"+SYMBOLE$(4):GET(X,Y-3)-(X+6,Y+6),MASSE1:CLS
150 PRESET(X,Y):DRAW "A2"+SYMBOLE$(4):GET(X-3,Y-6)-(X+6,Y),MASSE2:CLS
160 PRESET(X,Y):DRAW "A3"+SYMBOLE$(4):GET(X-6,Y-6)-(X,Y+3),MASSE3:CLS
170 FOR J=8 TO 192 STEP 8:FOR I=8 TO 312 STEP 8:PSET(I,J),2:NEXT I,J
180 GOSUB 410
190 ASCII=0:TOUCHE$=INKEY$:IF TOUCHE$="" THEN 210
200 ASCII=ASC(TOUCHE$):IF ASCII=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1))
210 GOSUB 410
```

```
220 IF ASCII=32 THEN 190
230 IF ASCII=82 THEN ROTATION=ROTATION+1:IF ROTATION>3 THEN ROTATION=0
240 IF ASCII=83 THEN GOTO 330
250 IF ASCII=43 THEN NUMERO=NUMERO+1:IF NUMERO>NOMBRE THEN NUMERO=1
260 IF ASCII=45 THEN NUMERO=NUMERO-1:IF NUMERO<1 THEN NUMERO=NOMBRE
270 IF ASCII=72 AND Y>8 THEN Y=Y-8
280 IF ASCII=75 AND X>8 THEN X=X-8
290 IF ASCII=77 THEN X=X+8
300 IF ASCII=80 THEN Y=Y+8
310 GOSUB 520
320 GOTO 180
330 FOR Y=8 TO 192 STEP 8:FOR X=8 TO 312 STEP 8
340 GOSUB 610
350 IF PIXELS=0 THEN PRESET(X,Y):GOTO 380
360 PSET(X,Y):IF PIXELS<4 THEN 380
370 PUT(X-1,Y-1),CONNEXION,PSET
380 NEXT X,Y
390 END
400 'Sous-programme d'affichage des symboles-----------------------------------
410 IF NUMERO=1 THEN PUT(X-1,Y-1),CONNEXION
420 IF NUMERO=2 AND (ROTATION=0 OR ROTATION=2) THEN PUT(X,Y),TRAITO
430 IF NUMERO=2 AND (ROTATION=1 OR ROTATION=3) THEN PUT(X,Y),TRAIT3
440 IF NUMERO=3 AND (ROTATION=0 OR ROTATION=2) THEN PUT(X+2,Y),POINTILLEO
450 IF NUMERO=3 AND (ROTATION=1 OR ROTATION=3) THEN PUT(X,Y+2),POINTILLE3
460 IF NUMERO=4 AND ROTATION=0 THEN PUT(X-6,Y),MASSEO
470 IF NUMERO=4 AND ROTATION=1 THEN PUT(X,Y-3),MASSE1
480 IF NUMERO=4 AND ROTATION=2 THEN PUT(X-3,Y-6),MASSE2
490 IF NUMERO=4 AND ROTATION=3 THEN PUT(X-6,Y-6),MASSE3
500 RETURN
510 'Sous-programme de traitement des sorties d'écran--------------------------
520 XMAX=312:YMAX=192
530 IF NUMERO=2 AND (ROTATION=0 OR ROTATION=2) THEN XMAX=304
540 IF NUMERO=2 AND (ROTATION=1 OR ROTATION=3) THEN YMAX=184
550 IF NUMERO=3 AND (ROTATION=0 OR ROTATION=2) THEN XMAX=304
560 IF NUMERO=3 AND (ROTATION=1 OR ROTATION=3) THEN YMAX=184
570 IF X>XMAX THEN X=XMAX
580 IF Y>YMAX THEN Y=YMAX
590 RETURN
600 'Sous-programme PIXEL de comptage du nombre de points allumés-------------
610 PIXELS=0
620 IF POINT(X+1,Y)<>0 THEN PIXELS=PIXELS+1
630 IF POINT(X+1,Y-1)<>0 THEN PIXELS=PIXELS+1
640 IF POINT(X,Y-1)<>0 THEN PIXELS=PIXELS+1
650 IF POINT(X-1,Y-1)<>0 THEN PIXELS=PIXELS+1
660 IF POINT(X-1,Y)<>0 THEN PIXELS=PIXELS+1
670 IF POINT(X-1,Y+1)<>0 THEN PIXELS=PIXELS+1
680 IF POINT(X,Y+1)<>0 THEN PIXELS=PIXELS+1
690 IF POINT(X+1,Y+1)<>0 THEN PIXELS=PIXELS+1
700 RETURN
```

Après simulation de la demande de sauvegarde du schéma à l'aide des touches [S] ou [Del] (ligne 240), le programme saute à la ligne 330. Or, que trouve-t-on à cette ligne 330 ?... La même boucle de balayage de l'écran que celle de la ligne 170 !... La seule différence avec celle-ci, c'est que l'on effectue un certain nombre de tests (lignes 350 à 370) avant de procéder à l'effacement des points de la grille.

Vous remarquerez que tous ces tests sont basés sur le nombre de PIXELS allumés adjacents au point de la grille (il, peut y en avoir jusqu'à 8...) et que c'est le sous-programme 610 qui se charge de les dénombrer. Si ce nombre est nul, c'est qu'il n'existe aucun symbole au voisinage immédiat du point de la grille et que celui-ci peut être effacé (ligne 350). Si ce nombre est inférieur à 4, c'est qu'il s'agit de tout autre chose

qu'un point de connexion (trait de liaison, pixels d'un symbole, etc...) et dans ce cas, il y a lieu de conserver le point de la grille (ligne 360). Par contre, si le nombre de **PIXELS** est égal ou supérieur à 4, c'est le signe incontestable que l'on a affaire à un point de connexion ou, ce qui est moins fréquent, à l'intérieur d'un symbole plein. Il ne reste plus alors dans ce cas qu'à dessiner le point de connexion.

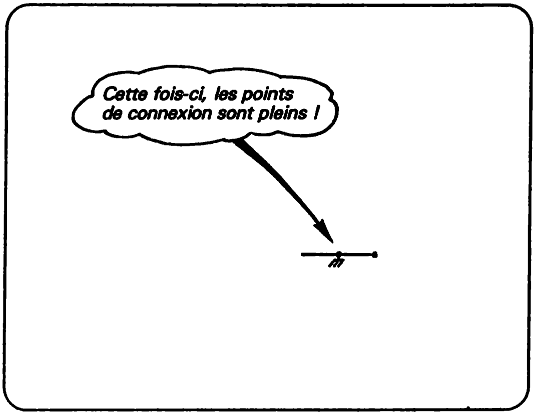


Fig. 11.9 : Schéma final obtenu après effacement de la grille

Et la reconstitution des pixels effacés dans tout cela, où a-t-elle lieu ?... Mais lors du dessin de la grille de points, ce que nous aurons l'occasion de voir en détail dans le programme final !

Dessinés en mode complémentaire, les points de connexion apparaissent troués. Il y a donc lieu de les redessiner correctement lors de l'effacement de la grille de points.

LE LOGICIEL GRAFELEC

La faisabilité du logiciel **GRAFELEC** étant désormais certaine (du moins on l'espère...), il ne nous reste plus qu'à assembler les divers programmes de simulation afin d'obtenir le programme complet de ce logiciel.

Voici donc pour commencer le programme de création de la bibliothèque de formes. Ne soyez pas trop surpris de sa longueur car il comporte 63 symboles différents :

```
10 DIM SYMBOLE$(63)
20 SYMBOLE$(1)="C3URD2L2U2":'CONNEXION
30 SYMBOLE$(2)="C3D8":'TRAIT0
40 SYMBOLE$(3)="C3R8":'TRAIT1
50 SYMBOLE$(4)="C3F8":'OBLIQUE0
60 SYMBOLE$(5)="C3G8":'OBLIQUE1
70 SYMBOLE$(6)="C3BD2D4":'POINTILLE0
80 SYMBOLE$(7)="C3BR2R4":'POINTILLE1
90 SYMBOLE$(8)="C3D6":'COUPURE0
100 SYMBOLE$(9)="C3R6":'COUPURE1
110 SYMBOLE$(10)="C3U6":'COUPURE2
120 SYMBOLE$(11)="C3L6":'COUPURE3
130 SYMBOLE$(12)="C3D3L3R6G3BU3G3BU3G3":'MASSE0
140 SYMBOLE$(13)="C3R3D3U6F3BL3F3BL3F3":'MASSE1
150 SYMBOLE$(14)="C3U3R3L6E3BD3E3BD3E3":'MASSE2
160 SYMBOLE$(15)="C3L3U3D6H3BR3H3BR3H3":'MASSE3
170 SYMBOLE$(16)="C3D4L2D8R4U8L2D3BD2D7":'FUSIBLE0
180 SYMBOLE$(17)="C3R4U2R8D4L8U2R3BR2R7":'FUSIBLE1
190 SYMBOLE$(18)="C3D8R3D16L3D8U8L3U16R3":'RESISTANCE0
200 SYMBOLE$(19)="C3R8U3R16D3R8L8D3L16U3":'RESISTANCE1
210 SYMBOLE$(20)="C3U4GR2FL4":'CURSEUR0
220 SYMBOLE$(21)="C3L4FU2ED4":'CURSEUR1
230 SYMBOLE$(22)="C3D4EL2HR4":'CURSEUR2
240 SYMBOLE$(23)="C3R4HD2GU4":'CURSEUR3
250 SYMBOLE$(24)="C3D6L3DR6UL3BD3L3DR6UL3D7":'CONDENSATEUR0
260 SYMBOLE$(25)="C3R6D3RU6LD3BR3D3RU6LD3R7":'CONDENSATEUR1
270 SYMBOLE$(26)="C3D5L3D2R6U2L3BD4L3DR6UL3D7":'CHIMIQUE0
280 SYMBOLE$(27)="C3R5D3R2U6L2D3BR4D3RU6LD3R7":'CHIMIQUE1
290 SYMBOLE$(28)="C3D7L3UR6DL3BD4L3U2R6D2L3D5":'CHIMIQUE2
300 SYMBOLE$(29)="C3R7D3LU6RD3BR4D3L2U6R2D3R5":'CHIMIQUE3
310 SYMBOLE$(30)="C3R3D32L3U32L3D32R3":'NOYAU0
320 SYMBOLE$(31)="C3U3R32D3L32D3R32U3":'NOYAU1
330 SYMBOLE$(32)="C3F2DG4LHU2ERF4D3G4LHU2ERF4DG
                 LRFDG4LHU2ERF4D3G4LHU2ERF4DG2":'INDUCTANCE0
340 SYMBOLE$(33)="C3E2RF4DGL2HUE4R3F4DGL2HUE4RF
                 DUERF4DGL2HUE4R3F4DGL2HUE4RF2":'INDUCTANCE1
350 SYMBOLE$(34)="C3G2DF4REU2HLG4D3F4REU2HLG4DF
                 RLGDF4REU2HLG4D3F4REU2HLG4DF2":'INDUCTANCE2
360 SYMBOLE$(35)="C3F2RE4UHL2GDF4R3E4UHL2GDF4RE
                 UDFRE4UHL2GDF4R3E4UHL2GDF4RE2":'INDUCTANCE3
370 SYMBOLE$(36)="C3R5D5U10FD8EU6FD4EU2FRU5D10U5R5":'DIODE0
380 SYMBOLE$(37)="C3D5R5L10R5DGR2FL4GR6FL8GR10L5D5":'DIODE1
390 SYMBOLE$(38)="C3R5U5D10U5RFU2ED4FU6ED8FU10D5R5":'DIODE2
400 SYMBOLE$(39)="C3D5L5R10GL8FR6GL4FR2GDR5L10R5D5":'DIODE3
410 SYMBOLE$(40)="C3E6H3F6UH5RF4UH3RF2UHREF3H6F3E6":'REDRESSEUR0
420 SYMBOLE$(41)="C3F6E3G6E3FRGDE2RG3DE4RG5DE6G3F6":'REDRESSEUR1
430 SYMBOLE$(42)="C3E6H3F6H3ERHUF2RH3UF4RH5UF6H3E6":'REDRESSEUR2
440 SYMBOLE$(43)="C3F6E3G6RE5DG4RE3DG2REDFE3G6E3F6":'REDRESSEUR3
450 SYMBOLE$(44)="C3R5D5U10FD8EU6FD4EU2FBU5RD10RBU5R4":'ZENER0
460 SYMBOLE$(45)="C3D4BR5DL10DBR5GR2FL4GR6FL8GR10L5D5":'ZENER1
470 SYMBOLE$(46)="C3R4BU5RD10RBU5FU2ED4FU6ED8FU10D5R5":'ZENER2
480 SYMBOLE$(47)="C3D5L5R10GL8FR6GL4FR2GBR5DL10DBR5D4":'ZENER3
490 SYMBOLE$(48)="C3E6H3F6UH5RF4UH3RF2UHRBH3EF6EBH3E5":'PONT0
500 SYMBOLE$(49)="C3F5BE3FG6FBE3RGDE2RG3DE4RG5DE6G3F6":'PONT1
510 SYMBOLE$(50)="C3E5BH3EF6EBH3RHUF2RH3UF4RH5UF6H3E6":'PONT2
520 SYMBOLE$(51)="C3F6E3G6RE5DG4RE3DG2REDBE3FG6FBE3F5":'PONT3
530 SYMBOLE$(52)="C3U3EUE2RER2BR2R2FRF2DFD2BD2D2GDG2L
                 GL2BL2L2HLH2UHU3R6U6RD12LU6R2E8G8F8":'TRANSISTOR0
540 SYMBOLE$(53)="C3L3HLH2UHU2BU2U2EUE2RER2BR2R2FRF2D
                 FD2BD2D2GDG2LGL3U6L6UR12DL6U2H8F8E8":'TRANSISTOR1
550 SYMBOLE$(54)="C3D3GDG2LGL2BL2L2HLH2UHU2BU2U2EUE2R
                 ER2BR2R2FRF2DFD3L6D6LU12RD6L2G8E8H8":'TRANSISTOR2
560 SYMBOLE$(55)="C3R3FRF2DFD2BD2D2GDG2LGL2BL2L2HLH2U
                 HU2BU2U2EUE2RER3D6R6DL12UR6D2F8H8G8":'TRANSISTOR1
```

```
570 SYMBOLE$(56)="C3BEBR3L2HU2":'PNP0
580 SYMBOLE$(57)="C3BHBU3D2GL2":'PNP1
590 SYMBOLE$(58)="C3BGBL3R2FD2":'PNP2
600 SYMBOLE$(59)="C3BFBD3U2ER2":'PNP3
610 SYMBOLE$(60)="C3BH3BU3D2GL2":'NPN0
620 SYMBOLE$(61)="C3BG3BL3R2FD2":'NPN1
630 SYMBOLE$(62)="C3BF3BD3U2ER2":'NPN2
640 SYMBOLE$(63)="C3BE3BR3L2HU2":'NPN3
650 FICHIER$="SYMBOLES.LIB":OPEN FICHIER$ FOR OUTPUT AS#1
660 FOR I=1 TO 63:WRITE#1,SYMBOLE$(I):NEXT:CLOSE#1
```

Quant au programme à proprement parler, le voici dans son intégralité :

```
10 DIM SYMBOLE$(63)
20 DIM CONNEXION(1),TRAIT0(3),TRAIT1(2)
30 DIM OBLIQUE0(7),OBLIQUE1(7)
40 DIM POINTILLE0(2),POINTILLE1(1)
50 DIM COUPURE0(2),COUPURE1(1),COUPURE2(2),COUPURE3(1)
60 DIM MASSE0(6),MASSE1(5),MASSE2(6),MASSE3(5)
70 DIM FUSIBLE0(9),FUSIBLE1(7)
80 DIM RESISTANCE0(17),RESISTANCE1(16)
90 DIM CURSEUR0(3),CURSEUR1(3),CURSEUR2(3),CURSEUR3(3)
100 DIM CONDENSATEUR0(9),CONDENSATEUR1(9)
110 DIM CHIMIQUE0(9),CHIMIQUE1(9),CHIMIQUE2(9),CHIMIQUE3(9)
120 DIM NOYAU0(17),NOYAU1(16)
130 DIM INDUCTANCE0(17),INDUCTANCE1(16),INDUCTANCE2(17),INDUCTANCE3(16)
140 DIM DIODE0(14),DIODE1(13),DIODE2(14),DIODE3(13)
150 DIM REDRESSEUR0(22),REDRESSEUR1(22),REDRESSEUR2(22),REDRESSEUR3(22)
160 DIM ZENER0(14),ZENER1(13),ZENER2(14),ZENER3(13)
170 DIM PONT0(22),PONT1(22),PONT2(22),PONT3(22)
180 DIM TRANSISTOR0(22),TRANSISTOR1(22),TRANSISTOR2(22),TRANSISTOR3(22)
190 DIM PNP0(1),PNP1(1),PNP2(1),PNP3(1)
200 DIM NPN0(1),NPN1(1),NPN2(1),NPN3(1)
210 DIM SCHEMA(660),MENU(160),DESSIN(28),GOMME(25)
220 DIM CHARGE(28),SAUVEGARDE(42),FIN(18)
230 KEY OFF:SCREEN 1:COLOR 8,1:CLS
240 GOSUB 760
250 GOSUB 1430
260 X=8:Y=8:NUMERO=1:CHOIX=1:NOMBRE=20
270 'Menu principal----------------------------------------------------
280 CLS
290 GET(0,167)-(319,199),SCHEMA
300 LINE(0,167)-(319,199),,B:LINE(1,168)-(318,198),0,BF
310 PUT(1,168),MENU,PRESET
320 IF CHOIX=1 THEN PUT(0,168),DESSIN,PSET
330 IF CHOIX=2 THEN PUT(64,168),GOMME,PSET
340 IF CHOIX=3 THEN PUT(120,168),CHARGE,PSET
350 IF CHOIX=4 THEN PUT(184,168),SAUVEGARDE,PSET
360 IF CHOIX=5 THEN PUT(280,168),FIN,PSET
370 TOUCHE$=INKEY$:IF TOUCHE$="" THEN 370
380 ASCII=ASC(TOUCHE$):IF ASCII=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1))
390 IF ASCII=75 THEN CHOIX=CHOIX-1:IF CHOIX<1 THEN CHOIX=5
400 IF ASCII=77 THEN CHOIX=CHOIX+1:IF CHOIX>5 THEN CHOIX=1
410 IF ASCII<>13 THEN 310
420 ON CHOIX GOTO 450,280,630,700
430 CLS:KEY ON:END
```

Vous y reconnaîtrez au passage les lignes de dimensionnement des tableaux graphiques (lignes 10 à 220), d'affectation des variables (lignes 260), de dessin de la fenêtre du menu principal (lignes 270 à 430)...

```
430 CLS:KEY ON:END
440 'Dessin du schéma électronique-------------------------------------------
450 PUT(0,167),SCHEMA,PSET
460 GOSUB 2450
470 GOSUB 1530
480 ASCII=0:TOUCHE$=INKEY$:IF TOUCHE$="" THEN 500
490 ASCII=ASC(TOUCHE$);IF ASCII=0 THEN ASCII=ASC(RIGHT$(TOUCHE$,1))
500 GOSUB 1530
510 IF ASCII=32 THEN 480
520 IF ASCII=27 THEN 2620
530 IF ASCII=82 THEN ROTATION=ROTATION+1:IF ROTATION>3 THEN ROTATION=0
540 IF ASCII=43 THEN NUMERO=NUMERO+1:IF NUMERO>NOMBRE THEN NUMERO=1
550 IF ASCII=45 THEN NUMERO=NUMERO-1:IF NUMERO<1 THEN NUMERO=NOMBRE
560 IF ASCII=72 AND Y>8 THEN Y=Y-8
570 IF ASCII=75 AND X>8 THEN X=X-8
580 IF ASCII=77 THEN X=X+8
590 IF ASCII=80 THEN Y=Y+8
600 GOSUB 2180
610 GOTO 470
620 'Chargement d'un schéma électronique-------------------------------------
630 LOCATE 24,2:INPUT;"Nom du fichier d'image: ",FICHIER$
640 IF FICHIER$="" THEN 290
650 PUT(0,167),SCHEMA,PSET
660 BLOAD FICHIER$
670 GOTO 290
680 RETURN
690 'Sauvegarde d'un schéma électronique-------------------------------------
700 LOCATE 24,2:INPUT;"Nom du fichier d'image: ",FICHIER$
710 IF FICHIER$="" THEN 290
720 PUT(0,167),SCHEMA,PSET
730 DEF SEG=&HB800:BSAVE FICHIER$,0,&H4000
740 GOTO 290
750 'Sous-programme de mise en tableau des symboles--------------------------
760 FICHIER$="SYMBOLES.LIB":OPEN FICHIER$ FOR INPUT AS#1
770 FOR I=1 TO 63:INPUT#1,SYMBOLE$(I):NEXT:CLOSE#1
780 PRESET(8,8):DRAW SYMBOLE$(1):GET(7,7)-(9,9),CONNEXION:CLS
790 PRESET(8,8):DRAW SYMBOLE$(2):GET(8,8)-(8,16),TRAIT0:CLS
800 PRESET(8,8):DRAW SYMBOLE$(3):GET(8,8)-(16,8),TRAIT1:CLS
810 PRESET(8,8):DRAW SYMBOLE$(4):GET(8,8)-(16,16),OBLIQUE0:CLS
820 PRESET(16,8):DRAW SYMBOLE$(5):GET(8,8)-(16,16),OBLIQUE1:CLS
830 PRESET(8,8):DRAW SYMBOLE$(6):GET(8,10)-(8,14),POINTILLE0:CLS
840 PRESET(8,8):DRAW SYMBOLE$(7):GET(10,8)-(14,8),POINTILLE1:CLS
850 PRESET(8,8):DRAW SYMBOLE$(8):GET(8,8)-(8,14),COUPURE0:CLS
860 PRESET(8,8):DRAW SYMBOLE$(9):GET(8,8)-(14,8),COUPURE1:CLS
870 PRESET(8,8):DRAW SYMBOLE$(10):GET(8,2)-(8,8),COUPURE2:CLS
880 PRESET(8,8):DRAW SYMBOLE$(11):GET(2,8)-(8,8),COUPURE3:CLS
890 PRESET(8,8):DRAW SYMBOLE$(12):GET(2,8)-(11,14),MASSE0:CLS
900 PRESET(8,8):DRAW SYMBOLE$(13):GET(8,5)-(14,14),MASSE1:CLS
910 PRESET(8,8):DRAW SYMBOLE$(14):GET(5,2)-(14,8),MASSE2:CLS
920 PRESET(8,8):DRAW SYMBOLE$(15):GET(2,2)-(8,11),MASSE3:CLS
930 PRESET(8,8):DRAW SYMBOLE$(16):GET(6,8)-(10,24),FUSIBLE0:CLS
940 PRESET(8,8):DRAW SYMBOLE$(17):GET(8,6)-(24,10),FUSIBLE1:CLS
950 PRESET(8,8):DRAW SYMBOLE$(18):GET(5,8)-(11,40),RESISTANCE0:CLS
960 PRESET(8,8):DRAW SYMBOLE$(19):GET(8,5)-(40,11),RESISTANCE1:CLS
```

... de dessin du schéma électronique (lignes 440 à 610), de chargement (lignes 620 à 670) ou de sauvegarde de ce dernier (lignes 690 à 740), suivies du sous-programme de mise en tableau des symboles (lignes 750 à 1410), de mise en tableau du menu (lignes 1420 à 1510)...

```
970 PRESET(8,8):DRAW SYMBOLE$(20):GET(6,4)-(10,8),CURSEUR0:CLS
980 PRESET(8,8):DRAW SYMBOLE$(21):GET(4,6)-(8,10),CURSEUR1:CLS
990 PRESET(8,8):DRAW SYMBOLE$(22):GET(6,8)-(10,12),CURSEUR2:CLS
1000 PRESET(8,8):DRAW SYMBOLE$(23):GET(8,6)-(12,10),CURSEUR3:CLS
1010 PRESET(8,8):DRAW SYMBOLE$(24):GET(5,8)-(11,24),CONDENSATEUR0:CLS
1020 PRESET(8,8):DRAW SYMBOLE$(25):GET(8,5)-(24,11),CONDENSATEUR1:CLS
1030 PRESET(8,8):DRAW SYMBOLE$(26):GET(5,8)-(11,24),CHIMIQUE0:CLS
1040 PRESET(8,8):DRAW SYMBOLE$(27):GET(8,5)-(24,11),CHIMIQUE1:CLS
1050 PRESET(8,8):DRAW SYMBOLE$(28):GET(5,8)-(11,24),CHIMIQUE2:CLS
1060 PRESET(8,8):DRAW SYMBOLE$(29):GET(8,5)-(24,11),CHIMIQUE3:CLS
1070 PRESET(8,8):DRAW SYMBOLE$(30):GET(5,8)-(11,40),NOYAU0:CLS
1080 PRESET(8,8):DRAW SYMBOLE$(31):GET(8,5)-(40,11),NOYAU1:CLS
1090 PRESET(8,8):DRAW SYMBOLE$(32):GET(4,8)-(10,40),INDUCTANCE0:CLS
1100 PRESET(8,8):DRAW SYMBOLE$(33):GET(8,6)-(40,12),INDUCTANCE1:CLS
1110 PRESET(8,8):DRAW SYMBOLE$(34):GET(6,8)-(12,40),INDUCTANCE2:CLS
1120 PRESET(8,8):DRAW SYMBOLE$(35):GET(8,4)-(40,10),INDUCTANCE3:CLS
1130 PRESET(8,8):DRAW SYMBOLE$(36):GET(8,3)-(24,13),DIODE0:CLS
1140 PRESET(8,8):DRAW SYMBOLE$(37):GET(3,8)-(13,24),DIODE1:CLS
1150 PRESET(8,8):DRAW SYMBOLE$(38):GET(8,3)-(24,13),DIODE2:CLS
1160 PRESET(8,8):DRAW SYMBOLE$(39):GET(3,8)-(13,24),DIODE3:CLS
1170 PRESET(8,24):DRAW SYMBOLE$(40):GET(8,8)-(24,24),REDRESSEUR0:CLS
1180 PRESET(8,8):DRAW SYMBOLE$(41):GET(8,8)-(24,24),REDRESSEUR1:CLS
1190 PRESET(8,24):DRAW SYMBOLE$(42):GET(8,8)-(24,24),REDRESSEUR2:CLS
1200 PRESET(8,8):DRAW SYMBOLE$(43):GET(8,8)-(24,24),REDRESSEUR3:CLS
1210 PRESET(8,8):DRAW SYMBOLE$(44):GET(8,3)-(24,13),ZENER0:CLS
1220 PRESET(8,8):DRAW SYMBOLE$(45):GET(3,8)-(13,24),ZENER1:CLS
1230 PRESET(8,8):DRAW SYMBOLE$(46):GET(8,3)-(24,13),ZENER2:CLS
1240 PRESET(8,8):DRAW SYMBOLE$(47):GET(3,8)-(13,24),ZENER3:CLS
1250 PRESET(8,24):DRAW SYMBOLE$(48):GET(8,8)-(24,24),PONT0:CLS
1260 PRESET(8,8):DRAW SYMBOLE$(49):GET(8,8)-(24,24),PONT1:CLS
1270 PRESET(8,24):DRAW SYMBOLE$(50):GET(8,8)-(24,24),PONT2:CLS
1280 PRESET(8,8):DRAW SYMBOLE$(51):GET(8,8)-(24,24),PONT3:CLS
1290 PRESET(8,16):DRAW SYMBOLE$(52):GET(8,8)-(24,24),TRANSISTOR0:CLS
1300 PRESET(16,24):DRAW SYMBOLE$(53):GET(8,8)-(24,24),TRANSISTOR1:CLS
1310 PRESET(24,16):DRAW SYMBOLE$(54):GET(8,8)-(24,24),TRANSISTOR2:CLS
1320 PRESET(16,8):DRAW SYMBOLE$(55):GET(8,8)-(24,24),TRANSISTOR3:CLS
1330 PRESET(8,8):DRAW SYMBOLE$(56):GET(9,4)-(12,7),PNP0:CLS
1340 PRESET(8,8):DRAW SYMBOLE$(57):GET(4,7)-(7,4),PNP1:CLS
1350 PRESET(8,8):DRAW SYMBOLE$(58):GET(4,9)-(7,12),PNP2:CLS
1360 PRESET(8,8):DRAW SYMBOLE$(59):GET(9,9)-(12,12),PNP3:CLS
1370 PRESET(8,8):DRAW SYMBOLE$(60):GET(2,2)-(5,5),NPN0:CLS
1380 PRESET(8,8):DRAW SYMBOLE$(61):GET(2,11)-(5,14),NPN1:CLS
1390 PRESET(8,8):DRAW SYMBOLE$(62):GET(11,11)-(14,14),NPN2:CLS
1400 PRESET(8,8):DRAW SYMBOLE$(63):GET(11,2)-(14,5),NPN3:CLS
1410 RETURN
1420 'Sous-programme de mise en tableau du menu--------------------------------
1430 LINE(0,167)-(319,199),,B
1440 LOCATE 22,2:PRINT"DESSIN  GOMME  CHARGE  SAUVEGARDE  FIN"
1450 GET(0,168)-(63,174),DESSIN
1460 GET(64,168)-(119,174),GOMME
1470 GET(120,168)-(183,174),CHARGE
1480 GET(184,168)-(279,174),SAUVEGARDE
1490 GET(280,168)-(319,174),FIN
1500 GET(1,168)-(318,175),MENU:CLS
1510 RETURN
1520 'Sous-programme d'affichage des symboles-------------------------------
1530 IF NUMERO=1 THEN PUT(X-1,Y-1),CONNEXION
1540 IF NUMERO=2 AND (ROTATION=0 OR ROTATION=2) THEN PUT(X,Y),TRAIT0
1550 IF NUMERO=2 AND (ROTATION=1 OR ROTATION=3) THEN PUT(X,Y),TRAIT1
1560 IF NUMERO=3 AND (ROTATION=0 OR ROTATION=2) THEN PUT(X,Y),OBLIQUE0
1570 IF NUMERO=3 AND (ROTATION=1 OR ROTATION=3) THEN PUT(X,Y),OBLIQUE1
1580 IF NUMERO=4 AND (ROTATION=0 OR ROTATION=2) THEN PUT(X,Y+2),POINTILLE0
1590 IF NUMERO=4 AND (ROTATION=1 OR ROTATION=3) THEN PUT(X+2,Y),POINTILLE1
```

```
1600 IF NUMERO=5 AND ROTATION=0 THEN PUT(X,Y),COUPURE0
1610 IF NUMERO=5 AND ROTATION=1 THEN PUT(X,Y),COUPURE1
1620 IF NUMERO=5 AND ROTATION=2 THEN PUT(X,Y-6),COUPURE2
1630 IF NUMERO=5 AND ROTATION=3 THEN PUT(X-6,Y),COUPURE3
1640 IF NUMERO=6 AND ROTATION=0 THEN PUT(X-6,Y),MASSE0
1650 IF NUMERO=6 AND ROTATION=1 THEN PUT(X,Y-3),MASSE1
1660 IF NUMERO=6 AND ROTATION=2 THEN PUT(X-3,Y-6),MASSE2
1670 IF NUMERO=6 AND ROTATION=3 THEN PUT(X-6,Y-6),MASSE3
1680 IF NUMERO=7 AND (ROTATION=0 OR ROTATION=2) THEN PUT(X-2,Y),FUSIBLE0
1690 IF NUMERO=7 AND (ROTATION=1 OR ROTATION=3) THEN PUT(X,Y-2),FUSIBLE1
1700 IF NUMERO=8 AND (ROTATION=0 OR ROTATION=2) THEN PUT(X-3,Y),RESISTANCE0
1710 IF NUMERO=8 AND (ROTATION=1 OR ROTATION=3) THEN PUT(X,Y-3),RESISTANCE1
1720 IF NUMERO=9 AND ROTATION=0 THEN PUT(X-2,Y-4),CURSEUR0
1730 IF NUMERO=9 AND ROTATION=1 THEN PUT(X-4,Y-2),CURSEUR1
1740 IF NUMERO=9 AND ROTATION=2 THEN PUT(X-2,Y),CURSEUR2
1750 IF NUMERO=9 AND ROTATION=3 THEN PUT(X,Y-2),CURSEUR3
1760 IF NUMERO=10 AND (ROTATION=0 OR ROTATION=2) THEN PUT(X-3,Y),CONDENSATEUR0
1770 IF NUMERO=10 AND (ROTATION=1 OR ROTATION=3) THEN PUT(X,Y-3),CONDENSATEUR1
1780 IF NUMERO=11 AND ROTATION=0 THEN PUT(X-3,Y),CHIMIQUE0
1790 IF NUMERO=11 AND ROTATION=1 THEN PUT(X,Y-3),CHIMIQUE1
1800 IF NUMERO=11 AND ROTATION=2 THEN PUT(X-3,Y),CHIMIQUE2
1810 IF NUMERO=11 AND ROTATION=3 THEN PUT(X,Y-3),CHIMIQUE3
1820 IF NUMERO=12 AND (ROTATION=0 OR ROTATION=2) THEN PUT(X-3,Y),NOYAU0
1830 IF NUMERO=12 AND (ROTATION=1 OR ROTATION=3) THEN PUT(X,Y-3),NOYAU1
1840 IF NUMERO=13 AND ROTATION=0 THEN PUT(X-4,Y),INDUCTANCE0
1850 IF NUMERO=13 AND ROTATION=1 THEN PUT(X,Y-2),INDUCTANCE1
1860 IF NUMERO=13 AND ROTATION=2 THEN PUT(X-2,Y),INDUCTANCE2
1870 IF NUMERO=13 AND ROTATION=3 THEN PUT(X,Y-4),INDUCTANCE3
1880 IF NUMERO=14 AND ROTATION=0 THEN PUT(X,Y-5),DIODE0
1890 IF NUMERO=14 AND ROTATION=1 THEN PUT(X-5,Y),DIODE1
1900 IF NUMERO=14 AND ROTATION=2 THEN PUT(X,Y-5),DIODE2
1910 IF NUMERO=14 AND ROTATION=3 THEN PUT(X-5,Y),DIODE3
1920 IF NUMERO=15 AND ROTATION=0 THEN PUT(X,Y),REDRESSEUR0
1930 IF NUMERO=15 AND ROTATION=1 THEN PUT(X,Y),REDRESSEUR1
1940 IF NUMERO=15 AND ROTATION=2 THEN PUT(X,Y),REDRESSEUR2
1950 IF NUMERO=15 AND ROTATION=3 THEN PUT(X,Y),REDRESSEUR3
1960 IF NUMERO=16 AND ROTATION=0 THEN PUT(X,Y-5),ZENER0
1970 IF NUMERO=16 AND ROTATION=1 THEN PUT(X-5,Y),ZENER1
1980 IF NUMERO=16 AND ROTATION=2 THEN PUT(X,Y-5),ZENER2
1990 IF NUMERO=16 AND ROTATION=3 THEN PUT(X-5,Y),ZENER3
2000 IF NUMERO=17 AND ROTATION=0 THEN PUT(X,Y),PONT0
2010 IF NUMERO=17 AND ROTATION=1 THEN PUT(X,Y),PONT1
2020 IF NUMERO=17 AND ROTATION=2 THEN PUT(X,Y),PONT2
2030 IF NUMERO=17 AND ROTATION=3 THEN PUT(X,Y),PONT3
2040 IF NUMERO=18 AND ROTATION=0 THEN PUT(X,Y),TRANSISTOR0
2050 IF NUMERO=18 AND ROTATION=1 THEN PUT(X,Y),TRANSISTOR1
2060 IF NUMERO=18 AND ROTATION=2 THEN PUT(X,Y),TRANSISTOR2
2070 IF NUMERO=18 AND ROTATION=3 THEN PUT(X,Y),TRANSISTOR3
2080 IF NUMERO=19 AND ROTATION=0 THEN PUT(X+1,Y-4),PNP0
2090 IF NUMERO=19 AND ROTATION=1 THEN PUT(X-4,Y-4),PNP1
2100 IF NUMERO=19 AND ROTATION=2 THEN PUT(X-4,Y+1),PNP2
2110 IF NUMERO=19 AND ROTATION=3 THEN PUT(X+1,Y+1),PNP3
2120 IF NUMERO=20 AND ROTATION=0 THEN PUT(X-6,Y-6),NPN0
2130 IF NUMERO=20 AND ROTATION=1 THEN PUT(X-6,Y+3),NPN1
2140 IF NUMERO=20 AND ROTATION=2 THEN PUT(X+3,Y+3),NPN2
2150 IF NUMERO=20 AND ROTATION=3 THEN PUT(X+3,Y-6),NPN3
2160 RETURN
```

... suivis à leur tour des sous-programmes d'affichage des symboles (lignes 1520 à 2160), de traitement des sorties d'écran (lignes 2170 à 2430), de dessin (lignes 2440 à 2600) et d'effacement de la grille de points (lignes 2610 à 2750) et pour finir, de comptage des points allumés (lignes 2760 à 2860) :

```
2170 'Sous-programme de traitement des sorties d'écran------------------------
2180 XMAX=312:YMAX=192
2190 IF NUMERO=2 AND (ROTATION=0 OR ROTATION=2) THEN YMAX=184
2200 IF NUMERO=2 AND (ROTATION=1 OR ROTATION=3) THEN XMAX=304
2210 IF NUMERO=3 THEN XMAX=304:YMAX=184
2220 IF NUMERO=7 AND (ROTATION=0 OR ROTATION=2) THEN YMAX=176
2230 IF NUMERO=7 AND (ROTATION=1 OR ROTATION=3) THEN XMAX=296
2240 IF NUMERO=8 AND (ROTATION=0 OR ROTATION=2) THEN YMAX=160
2250 IF NUMERO=8 AND (ROTATION=1 OR ROTATION=3) THEN XMAX=280
2260 IF NUMERO=10 AND (ROTATION=0 OR ROTATION=2) THEN YMAX=176
2270 IF NUMERO=10 AND (ROTATION=1 OR ROTATION=3) THEN XMAX=296
2280 IF NUMERO=11 AND (ROTATION=0 OR ROTATION=2) THEN YMAX=176
2290 IF NUMERO=11 AND (ROTATION=1 OR ROTATION=3) THEN XMAX=296
2300 IF NUMERO=12 AND (ROTATION=0 OR ROTATION=2) THEN YMAX=160
2310 IF NUMERO=12 AND (ROTATION=1 OR ROTATION=3) THEN XMAX=280
2320 IF NUMERO=13 AND (ROTATION=0 OR ROTATION=2) THEN YMAX=160
2330 IF NUMERO=13 AND (ROTATION=1 OR ROTATION=3) THEN XMAX=280
2340 IF NUMERO=14 AND (ROTATION=0 OR ROTATION=2) THEN XMAX=296
2350 IF NUMERO=14 AND (ROTATION=1 OR ROTATION=3) THEN YMAX=176
2360 IF NUMERO=15 THEN XMAX=296:YMAX=176
2370 IF NUMERO=16 AND (ROTATION=0 OR ROTATION=2) THEN XMAX=296
2380 IF NUMERO=16 AND (ROTATION=1 OR ROTATION=3) THEN YMAX=176
2390 IF NUMERO=17 THEN XMAX=296:YMAX=176
2400 IF NUMERO=18 THEN XMAX=296:YMAX=176
2410 IF X>XMAX THEN X=XMAX
2420 IF Y>YMAX THEN Y=YMAX
2430 RETURN
2440 'Sous-programme de dessin de la grille de points-------------------------
2450 FOR J=8 TO 192 STEP 8:FOR I=8 TO 312 STEP 8
2460 GOSUB 2770
2470 IF PIXELS MOD 2=1 THEN PSET(I,J),1:GOTO 2590
2480 IF PIXELS<>8 THEN PSET(I,J),2:GOTO 2590
2490 IF POINT(I-2,J-1)<>0 OR POINT(I+2,J+1)<>0 THEN PSET(I,J),1:GOTO 2590
2500 IF POINT(I+2,J)<>0 THEN PRESET(I+1,J):PIXELS=PIXELS-1
2510 IF POINT(I+2,J-2)<>0 THEN PRESET(I+1,J-1):PIXELS=PIXELS-1
2520 IF POINT(I,J-2)<>0 THEN PRESET(I,J-1):PIXELS=PIXELS-1
2530 IF POINT(I-2,J-2)<>0 THEN PRESET(I-1,J-1):PIXELS=PIXELS-1
2540 IF POINT(I-2,J)<>0 THEN PRESET(I-1,J):PIXELS=PIXELS-1
2550 IF POINT(I-2,J+2)<>0 THEN PRESET(I-1,J+1):PIXELS=PIXELS-1
2560 IF POINT(I,J+2)<>0 THEN PRESET(I,J+1):PIXELS=PIXELS-1
2570 IF POINT(I+2,J+2)<>0 THEN PRESET(I+1,J+1):PIXELS=PIXELS-1
2580 IF PIXELS MOD 2=0 THEN PSET(I,J),1 ELSE PSET(I,J),2
2590 NEXT I,J
2600 RETURN
2610 'Sous-programme d'effacement de la grille de points----------------------
2620 FOR J=8 TO 192 STEP 8:FOR I=8 TO 312 STEP 8
2630 GOSUB 2770
2640 IF PIXELS=0 OR PIXELS=6 THEN PRESET(I,J):GOTO 2740
2650 PSET(I,J):IF PIXELS<4 THEN 2740
2660 IF POINT(I+2,J)<>0 THEN PSET(I+1,J)
2670 IF POINT(I+2,J-2)<>0 THEN PSET(I+1,J-1)
2680 IF POINT(I,J-2)<>0 THEN PSET(I,J-1)
2690 IF POINT(I-2,J-2)<>0 THEN PSET(I-1,J-1)
2700 IF POINT(I-2,J)<>0 THEN PSET(I-1,J)
2710 IF POINT(I-2,J+2)<>0 THEN PSET(I-1,J+1)
2720 IF POINT(I,J+2)<>0 THEN PSET(I,J+1)
2730 IF POINT(I+2,J+2)<>0 THEN PSET(I+1,J+1)
2740 NEXT I,J
2750 GOTO 290
2760 'Sous-programme de comptage des points allumés---------------------------
2770 PIXELS=0
2780 IF POINT(I+1,J)<>0 THEN PIXELS=PIXELS+1
```

```
2790 IF POINT(I+1,J-1)<>0 THEN PIXELS=PIXELS+1
2800 IF POINT(I,J-1)<>0 THEN PIXELS=PIXELS+1
2810 IF POINT(I-1,J-1)<>0 THEN PIXELS=PIXELS+1
2820 IF POINT(I-1,J)<>0 THEN PIXELS=PIXELS+1
2830 IF POINT(I-1,J+1)<>0 THEN PIXELS=PIXELS+1
2840 IF POINT(I,J+1)<>0 THEN PIXELS=PIXELS+1
2850 IF POINT(I+1,J+1)<>0 THEN PIXELS=PIXELS+1
2860 RETURN
```

Le résultat ?... En voici un exemple qui n'a demandé en tout et pour tout que 5 minutes de travail (à condition de compiler(*) le programme, sinon...) :

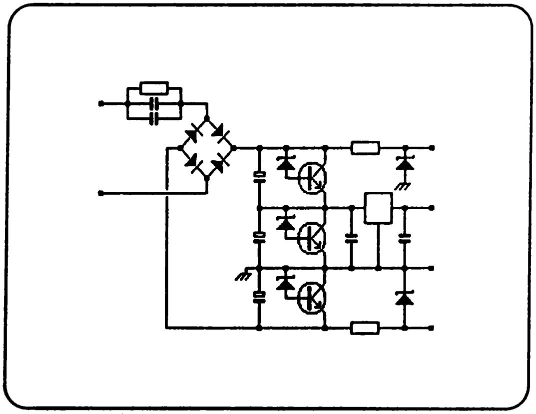


Fig. 11.10 : Un exemple de schéma électronique obtenu avec le logiciel GRAFELEC

(*) Que ceux pour qui la notion de programme compilé est nouvelle veuillent bien se reporter à l'ouvrage de H. Lilen « Pratique du Basic compilé ».

CHAPITRE XII

BRUITAGE ET MUSIQUE

En attendant la prochaine génération de micro-ordinateurs à synthèse de parole, voici de quoi donner de la voix à un compagnon qui a su se taire depuis le début de cet ouvrage. Du simple bip sonore à la musique d'accompagnement, vos programmes vont désormais exciter agréablement l'organe de l'ouïe de ceux qui les utiliseront (sauf en cas de fausse note, s'entend...), les rendant ainsi bien plus vivants et, de ce fait, plus attractifs d'emploi.

Menu du chapitre 12

Bruitage d'un programme
Hauteur et durée des notes
Transcription d'une partition musicale
Le sous-langage musical PLAY
Hey Jude

Quel est votre choix : ■

BRUITAGE D'UN PROGRAMME

Il peut arriver dans un programme que certaines situations nécessitent un rappel à l'ordre de l'opérateur, comme par exemple lorsque ce dernier cherche à sauvegarder un fichier sur une disquette protégée contre l'écriture. Or, le meilleur moyen d'attirer son attention est de produire un bip sonore.

BEEP

Pour cela, il vous suffit d'introduire dans le programme l'instruction **BEEP** (*), comme dans l'exemple suivant :

L'instruction BEEP produit un bip sonore

```
10 CLS:LOCATE 1,1:INPUT"MOT DE PASSE: ",MOT$
20 IF MOT$<> "TEA FOR TWO" THEN BEEP:GOTO 10
```

(*) La fiche détaillée de l'instruction BEEP figure à la page 256 du volume 2 de Pratique des IBM-PC et compatibles.

Remarque : ***L'instruction BEEP a le même effet qu'un PRINT CHR$ (7) qui à son tour est équivalent à un Ctrl – G.***

D'accord, ce n'est pas très harmonieux, mais c'est simple à mettre en œuvre !...

> **L'instruction BEEP produit un bip sonore analogue à celui qu'on obtient en mode direct en appuyant sur Ctrl – G.**

HAUTEUR ET DURÉE DES NOTES

SOUND

Heureusement, il est possible de faire mieux en se servant de l'instruction **SOUND** (*) qui permet de doser à la fois la hauteur de la note et sa durée, ce que ne permettait pas la précédente instruction **BEEP**.

Sachant d'une part que la syntaxe d'écriture de cette instruction est...

SOUND *hauteur, durée*

... et que d'autre part la *hauteur* peut varier de 37 à 32767 et la *durée* de – 65535 à + 65535 (le signe – n'étant pas pris en compte...), vous allez pouvoir procéder aux quelques essais que voici :

```
10 SOUND 440,4.4
```

Tiens, revoici le bip sonore !

```
10 SOUND 800,4.4
```

Çà, c'est un LA !

```
10 SOUND 440,176
```

Quant à ce LA là, il dure 10 secondes

Des quelques essais qui précèdent, il est possible d'en déduire que le bip sonore que procure l'instruction **BEEP** correspond à une hauteur de note égale à 800 hertz (essai n° 1), que la hauteur du LA est égale à 440 hertz (ce qui se vérifie avec un diapason...) et qu'une seconde de durée s'obtient avec une valeur de 17.6 (essai n° 3).

Fort de ces constations et sachant que chaque demi-ton de la gamme musicale s'obtient en multipliant la hauteur du précédent par $\sqrt[12]{2}$ (1.059463...), nous voici à même d'établir la hauteur des notes des quatre octaves couramment utilisées en musique :

(*) La fiche détaillée de l'instruction SOUND figure à la page 262 du volume 2 de Pratique des IBM-PC.

Notes	Octave n° 1	Octave n° 2	Octave n° 3	Octave n° 4
DO		262	523	1047
DO#		277	554	1109
RE		294	587	1175
RE#		311	622	1245
MI		330	659	1319
FA	175	349	698	
FA#	185	370	740	
SOL	196	392	784	
SOL#	208	415	831	
LA	220	440	880	
LA#	233	466	932	
SI	247	494	988	

Tableau 12.1 — Hauteur des notes de la gamme musicale

Ce point étant acquis, il reste à déterminer la durée des notes. Or vous savez (tout du moins pour ceux qui jouent d'un instrument...) que la durée d'une note dépend à la fois de sa nature (blanche, noire, croche, etc...) et du tempo du passage musical (largo, presto, prestissimo, etc...). C'est pourquoi vous allez obtenir le tableau de durée des notes que voici :

Notes	Tempo (1 seconde = 17.6)						
	60	80	100	120	140	160	180
Ronde	70.4	52.8	42.2	35.2	30.2	26.4	23.5
Blanche pointée	52.8	39.6	31.7	26.4	22.6	19.8	17.6
Blanche	35.2	26.4	21.1	17.6	15.1	13.2	11.7
Noire pointée	26.4	19.8	15.8	13.2	11.3	9.9	8.8
Noire	17.6	13.2	10.6	8.8	7.5	6.6	5.9
Croche pointée	13.2	9.9	7.9	6.6	5.7	5.0	4.4
Croche	8.8	6.6	5.2	4.4	3.8	3.3	2.9
Double croche	4.4	3.3	2.6	2.2	1.9	1.6	1.5
Triple croche	2.2	1.6	1.3	1.1	0.9	0.8	0.7
Quadruple croche	1.1	0.8	0.7	0.6	0.5	0.4	0.3

Tableau 12.7 — Durée des notes selon le tempo

Vous noterez au travers de ce tableau que plus le tempo est rapide et plus la durée de la note est courte. C'est ainsi qu'une noire va durer 8.8/17.6 seconde, soit 1/2 seconde, lorsqu'elle est jouée sur un tempo de 120 (en fait, ce nombre correspond au nombre de coups par minute d'un métronome...), tandis qu'elle va durer le double, soit 1 seconde entière, lorsqu'elle est jouée sur un tempo de 60.

Et les silences dans tout cela, comment les obtient-on ?... Mais en programmant une note de hauteur suffisamment élevée pour que l'oreille ne l'entende pas !... Sachant qu'en moyenne, celle-ci « débraye » au-delà de 16 000 à 17 000 hertz, il vous suffit de programmer vos silences avec la valeur 32 767 pour que ceux-ci soient totalement inaudibles (sauf pour votre chien, mais cela, c'est une autre histoire...).

Les silences étant assujettis aux mêmes règles de durée que les notes musicales, nous allons retrouver un tableau de codage sensiblement identique au précédent :

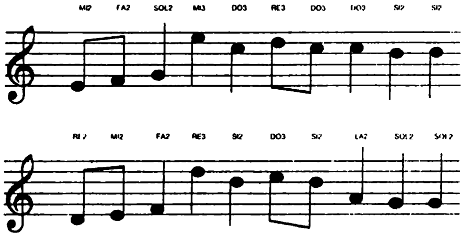
Silences (h = 32767)	Tempo (1 seconde = 17.6)						
	60	80	100	120	140	160	180
Pause	70.4	52.8	42.2	35.2	30.2	26.4	23.5
Demi-pause	35.2	26.4	21.1	17.6	15.1	13.2	11.7
Soupir	17.6	13.2	10.6	8.8	7.5	6.6	5.9
Demi-soupir	8.8	6.6	5.2	4.4	3.8	3.3	2.9
Quart de soupir	4.4	3.3	2.6	2.2	1.9	1.6	1.4

Tableau 12.3 — Durée des silences

> **L'instruction SOUND permet d'obtenir de véritables notes musicales dont la hauteur, la durée et le tempo dépendent des valeurs des tableaux 12.1 à 12.3.**

TRANSCRIPTION D'UNE PARTITION MUSICALE

Après la théorie, il est temps de passer à la pratique !... Nous allons nous initier à la transcription d'une partion musicale en débutant par quelque chose de simple, à savoir le début de « It's a small world » que l'on peut entendre dans les parcs d'attraction de Walt Disney :



Normalement si vous ne commettez pas d'erreur, vous devriez aboutir au petit programme que voici :

Codage des 20 premières notes de «It's a small world»

```
10 FOR NOTES=1 TO 20
20 READ HAUTEUR,DUREE
30 SOUND HAUTEUR,DUREE
40 NEXT
50 DATA 330,4.4,349,4.4,392,8.8,659,8.8
60 DATA 523,8.8,587,4.4,523,4.4,523,8.8
70 DATA 494,8.8,494,8.8,294,4.4,330,4.4
80 DATA 349,8.8,587,8.8,494,8.8,523,4.4
90 DATA 494,4.4,440,8.8,392,8.8,392,8.8
```

Exécutez-le !... Que constatez-vous ?... Que lorsque deux notes contigües ont la même hauteur (telles que les 7e et 8e par exemple...), rien ne les distingue l'une de l'autre car l'instruction **SOUND** les lie ensemble. Cela tient à ce que cette instruction fonctionne en mode d'arrière-plan, c'est-à-dire que pendant qu'elle joue une note, le programme continue son exécution. De ce fait, le temps mort qui devrait exister à chaque lecture des valeurs de **HAUTEUR** et de **DUREE** n'existe plus, et l'instruction **SOUND** enchaîne sans interruption avec la note suivante.

Ce fonctionnement en mode d'arrière-plan, loin de constituer un avantage, se traduit par cet inconvénient majeur. En conséquence, nous allons abandonner l'instruction **SOUND** pour une nouvelle instruction bien plus performante dès lors que nous cherchons à produire de la musique. Par contre, si on se limite à quelques bruitages, comme celui de cette sirène de pompier, l'instruction **SOUND** est parfaitement adaptée à cette fonction :

```
10 SOUND 523,8.8
20 SOUND 32767,2.2
30 SOUND 392,8.8
40 SOUND 32767,2.2
50 GOTO 10
```

Sirène d'une voiture de pompiers

L'emploi de l'instruction SOUND doit être limité à la production de bruitages.

LE SOUS-LANGAGE MUSICAL PLAY

PLAY

Nettement plus performante que la précédente, l'instruction **PLAY**(*) est à elle seule un véritable sous-langage musical. D'une syntaxe similaire à celle du sous-langage graphique...

PLAY *« chaîne de caractères »*

... celui-ci nécessite une chaîne de caractères dans laquelle on place les différentes lettres-code de commande.

(*) La fiche détaillée de l'instruction PLAY figure à la page 258 du volume 2 de Pratique des IBM-PC.

Hauteur des notes

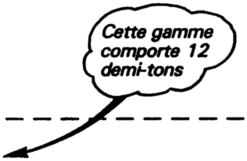
Premier avantage de ce sous-langage : il est inutile de rechercher la fréquence d'une note pour la jouer. Il suffit de préciser avec la lettre-code appropriée s'il s'agit d'un DO, d'un RÉ, d'un MI, etc..., comme indiqué par le tableau suivant :

Note	Lettre-code
DO	C
RE	D
MI	E
FA	F
SOL	G
LA	A
SI	B

Tableau 12.4 — Lettres-code des notes de musique

Mieux encore !... Les dièses s'obtiennent en faisant suivre la lettre-code de la note du caractère ... dièse (#). On ne peut faire plus clair !

Vous allez d'ailleurs pouvoir vérifier tout cela avec l'exemple que voici :




```
10 GAMME$="CC#DD#EFF#GG#AA#B"
20 PLAY GAMME$
```

Remarque : *Veillez bien à ne placer dans la chaîne de caractères que des lettres-code acceptables, sans quoi l'ordinateur vous interrompt votre morceau musical par un ILLEGAL FUNCTION IN...*

La hauteur de chaque note dépend de la lettre-code employée (de A à G) auquel on rajoute le signe # pour les dièses.

Octaves et changement d'octaves

La gamme que vous venez de jouer ne vous paraît-elle pas haut perchée ?... En fait, c'est normal car en l'absence d'indication de numéro d'octave, l'instruction PLAY joue dans l'octave n° 4 (c'est l'octave sélectionnée par défaut).

Pour sélectionner une octave différente, il vous suffit d'employer la lettre-code O suivie du numéro de l'octave. Or, sachant que vous disposez au total de 7 octaves numérotées de 0 à 6, vous allez pouvoir jouer les 84 demi-tons (7 × 12) disponibles grâce au petit programme que voici :

Avec l'instruction PLAY, vous disposez de 7 octaves

```
10 FOR OCTAVE=0 TO 6
20 GAMME$="O=OCTAVE;CC#DD#EFF#GG#AA#B"
30 PLAY GAMME$
40 NEXT
```

Vous noterez l'écriture qui consiste à affecter une variable (**OCTAVE** en l'occurrence...) à **la lettre-code O, le tout étant délimité par un point-virgule.**

Bien que **pratique, cette écriture n'est pas la seule que vous puissiez utiliser. Vous pouvez lui préférer celle qui consiste à faire suivre la lettre-code O du numéro de l'octave, comme dans l'exemple ci-dessous :**

Le choix de l'octave se fait ici !

```
10 GAMME$="O2CC#DD#EFF#GG#AA#B"
20 PLAY GAMME$
```

Remarque : ***Du fait qu'une octave reste active tant que l'on n'en spécifie pas une autre, il est inutile de répéter le numéro d'octave lorsque plusieurs notes consécutives sont jouées sur la même octave.***

Ainsi donc, il suffit de préciser le nouveau numéro d'octave chaque fois que l'on souhaite monter ou descendre d'octave. Toutefois, il existe une méthode plus élégante pour changer d'octave. Il suffit d'employer le signe plus grand que (>) ou plus petit que (<) pour passer d'une octave à celle immédiatement supérieure ou inférieure. En voici un exemple de démonstration :

Passage à l'octave immédiatement supérieure

```
10 GAMME1$="O2 CDEFGAB>C"
20 GAMME2$="O3 C<BAGFEDC"
30 PLAY GAMME1$+GAMME2$
```

Vous noterez que l'instruction PLAY accepte la concaténation de chaînes, comme c'est le cas dans l'exemple ci-dessus. Cette facilité qui nous est offerte nous simplifiera grandement la vie dans le cas d'un refrain.

Remarque : ***Vous pouvez placer autant de signes (>) ou (<) à la suite les uns des autres que vous souhaitez sauter d'octaves.***

L'instruction PLAY dispose de 7 octaves (numérotées de 0 à 6) que l'on sélectionne avec la lettre-code O ou avec les signes > ou <.

Durée des notes

La durée des notes est spécifiée par la lettre-code L suivie d'un nombre compris entre 1 (durée la plus grande) et 64 (durée la plus courte). Vous allez donc pouvoir programmer toute note (sauf les notes pointées) de la ronde à la croche, en accord avec le tableau que voici :

Note	Durée
Ronde	1
Blanche	2
Noire	4
Croche	8
Double croche	16
Triple croche	32
Quadruple croche	64

Tableau 12.5 – Durée des notes

Afin de vérifier tout cela, essayez donc de jouer la gamme montante précédente en faisant varier la durée des notes de 1 (2^0) à 64 (2^6) par pas de puissance de deux, comme dans l'exemple suivant :

```
10 FOR I=0 TO 6:DUREE=2^I
20 GAMME$="L=DUREE;O2 CDEFGAB>C"
30 PLAY GAMME$
40 NEXT
```

La lettre-code L règle la durée des notes

Vous remarquerez au passage que la durée des notes reste active tant qu'on n'en spécifie pas une autre et que cette durée est égale à 4 par défaut (ce qui correspond à une noire).

Bien que simple d'emploi, cette écriture est inappropriée à la production d'un morceau de musique dans lequel les notes n'ont pas forcément toutes la même longueur. Vous lui préférerez dans ce cas l'écriture qui consiste à abandonner la lettre-code L et à placer la durée immédiatement à la suite de la note, comme dans l'essai ci-après :

```
10 ESSAI$="O2 E8F8G4>E4C4D8C8C4<B4B4D8E8F4>D4<B4>C8<B8A4G4G4"
20 PLAY ESSAI$
```

Vous avez reconnu le début de l'air « Its a small world » que nous avions tenté de coder précédemment en nous servant de l'instruction SOUND. Cette fois-ci les notes sont bien détachées les unes des autres.

> **La durée des notes se code en faisant suivre la lettre-code de hauteur (A à G) du nombre correspondant à la durée (1 à 64).**

Notes pointées

Le sous-langage **PLAY** traite différemment les notes pointées de celles qui ne le sont pas. En effet, une note pointée voit sa durée augmentée de 50 % dês lors qu'elle est suivie d'un point (comme sur une partition musicale...).
Dans ces conditions, il vous suffit de rajouter un point à la suite de l'indication de durée de la note, comme cela est indiqué par le tableau suivant :

Note	Durée
Blanche pointée	2.
Noire pointée	4.
Croche pointée	8.

Tableau 12.6 — Durée des notes pointées

En guise d'essai, sortez donc votre chronomètre et vérifiez que la note pointée suivante (c'est une blanche...) dure bien 1,5 seconde, soit 50 % de temps de plus qu'une blanche normale :

Le point augmente de 50 % la durée des notes pointées

```
10 PLAY "O2 A2."
```

Un point placé à la suite de l'indication de durée d'une note augmente de 50 % la durée de celle-ci.

Les silences musicaux

Utilisant la lettre-code **P**, les silences musicaux se traitent de la même façon que les notes, à savoir qu'il faut faire suivre cette lettre-code d'un chiffre compris entre 1 et 16, comme cela est précisé dans le tableau suivant :

Silence	Durée
Pause	1
Demi-pause	2
Soupir	4
Demi-soupir	8
Quart de soupir	16

Tableau 12.7 — Durée des silences

> **Les silences s'obtiennent à l'aide de la lettre-code P suivie d'un nombre compris entre 1 et 16.**

Le tempo

Comme vous avez pu vous en rendre compte avec l'instruction **SOUND**, le codage du tempo allait de pair avec celui de la durée de chaque note, ce qui rendait toute modification du tempo difficile à réaliser puisqu'il fallait recalculer pour chaque note sa nouvelle durée (vous parlez d'un travail...).

Or, cela n'est plus le cas avec l'instruction **PLAY** car le tempo se détermine indépendamment de la durée des notes. Fixé par défaut à 120 coups par minute, celui-ci peut être modifié à votre gré à l'aide de la lettre-code **T** suivie d'un nombre compris entre 32 et 255.

Ainsi, pour jouer l'essai précédent sur des tempos différents, il vous suffit d'écrire :

Fixation du tempo de cet essai musical

```
10 FOR TEMPO=60 TO 180 STEP 60
20 PLAY"T=TEMPO;"
30 ESSAI$="O2 E8F8G4>E4C4D8C8C4<B4B4D8E8F4>D4<B4>C8<B8A4G4G4"
40 PLAY ESSAI$
50 NEXT
```

Vous remarquerez la façon d'introduire le tempo musical à l'aide d'un **PLAY** séparé. Bien que cela ne soit pas obligatoire, cela vous facilite la programmation du morceau de musique dans la mesure où le tempo est toujours le même du début à la fin (ce qui est généralement le cas).

> **Le tempo se fixe avec la lettre-code T suivie d'un nombre compris entre 32 (pour le plus lent) et 255 (pour le plus rapide).**

L'articulation des notes

Qu'est-ce que l'articulation des notes ?... C'est ce qui fait qu'une note est jouée pendant tout ou partie du temps qui lui est allouée. Or, c'est ce qui précisément distingue l'instruction **PLAY** de l'instruction **SOUND**, cette dernière étant incapable de changer quoi que ce soit à l'articulation des notes qui étaient jouées pendant la durée totale du temps alloué, alors qu'au contraire, l'instruction **PLAY** possède 3 modes de fonctionnement :

- le mode *legato* (lettres-code ML) avec lequel les notes sont jouées pendant la totalité du temps qui leur est imparti.
- le mode *normal* (lettres-codes MN) avec lequel les notes sont jouées pendant les 7/8e du temps alloué. C'est le mode sélectionné par défaut.
- le mode *staccato* (lettres-codes MS) avec lequel les notes sont jouées pendant les 3/4 du temps qui leur est accordé.

Pour bien distinguer ces 3 modes l'un de l'autre, introduisez-les donc à tour de rôle dans l'instruction PLAY pour voir (ou plutôt écouter...) l'effet produit :

```
10 ESSAI$="O2 E8F8G4>E4C4D8C8C4<B4B4D8E8F4>D4<B4>C8<B8A4G4G4"
20 PLAY "T140"
30 PLAY "ML"+ESSAI$
40 PLAY "MN"+ESSAI$
50 PLAY "MS"+ESSAI$
```

Remarquez comme le mode staccato « martèle » les notes comparé au mode normal qui, au contraire, les lient les unes aux autres. C'est cela, l'articulation des notes de **l'instruction PLAY.**

L'instruction PLAY dispose de 3 modes d'articulation des notes, le mode legato (ML), le mode normal (MN) et le mode staccato (MS).

HEY JUDE

Histoire de terminer en beauté ce chapitre sur la musique, je vous propose de transcrire la chanson « Hey Jude » des Beatles qui traîne encore dans quelque recoin de la mémoire...

```
10 KEY OFF:SCREEN 0:COLOR 15,9:CLS
20 LOCATE 13,35:PRINT"HEY JUDE..."
30 PLAY"T80O2"
40 PLAY"G4MLE2MNE8E8G8A8D2P4D8E8F4>MLC4MNC8C8<B8G8"
50 PLAY"MLA8G16MNF16E2P8G8A8A4A8>D8C16<MLB16MNB16>C16<A8G4.P8C8D8E8MLA8"
60 PLAY"MNG4.G8MLF8MNE4<B8>C8MLC8C4MNC8P8
70 PLAY"G4MLE2MNE8E8G8A8D2P4D8E8F4>MLC4MNC8C8<B8G8"
80 PLAY"MLA8G16MNF16E2P8G8A8A4A8>D8C16<MLB16MNB16>C16<A8G4.P8C8D8E8MLA8"
90 PLAY"MNG4.G8MLF8MNE4<B8>C8MLC8C4MNC2P8
100 PLAY"F8C8>C8<B16MLA16MNA8G8G8F16MLA16MNA8>C8<MLF8MNF4>C8<MLA8MNA8G8F8MLG8"
110 PLAY"MNG4A8MLG8MNG4F8MLE8E8MND8MLC4MNC2P8C8>C8<A16MLA16MNA8G8G8F16MLA16"
120 PLAY"MNA4>C8<MLA8MNA4>C8<MLF8MNF4>C8<MLA8MNA8G8F8MLG8MNG4A8MLG8MNG4F8MLE8"
130 PLAY"E8D8MNC4P8C8G8A8A#8A8B4B8>C8D4D2P4<G4MLE2MNE8E8G8A8"
140 PLAY"MND2P4D8E8F4ML>C4MNC8C8<B8G8MLA8G16MNF16E2P8G8"
150 PLAY"A8A4A8>D8C16ML<B16MNB16>C16<A8G4.P8C8D8E8MLA8MNG4.G8MLF8MNE4<B8"
160 PLAY">C8MLC8MNC2"
170 PLAY"F8C8>C8<B16MLA16MNA8G8G8F16MLA16MNA8>C8<MLF8MNF4>C8<MLA8MNA8G8F8MLG8"
180 PLAY"MNG4A8MLG8MNG4F8MLE8E8MND8MLC4MNC2P8C8>C8<A16MLA16MNA8G8G8F16MLA16"
190 PLAY"MNA4>C8<MLA8MNA4>C8<MLF8MNF4>C8<MLA8MNA8G8F8MLG8MNG4A8MLG8MNG4F8MLE8"
200 PLAY"E8D8MNC4P8C8G8A8A#8A8B4B8>C8D4D2P4<G4MLE2MNE8E8G8A8"
210 PLAY"MND2P4D8E8F4ML>C4MNC8C8<B8G8MLA8G16MNF16E2P8G8"
220 PLAY"A8A4A8>D8C16ML<B16MNB16>C16<A8G4.P8C8D8E8MLA8MNG4.G8MLF8MNE4<B8"
230 PLAY">C8MLC8MNC2P4C2E4G4>D16C16D8C2.D16C16D8C2<A4G2.P4"
240 PLAY"C2E4G4>D16C16D8C2.D16C16D8C2<A4G2."
```

Exercices sur le chapitre 12

1. Peut-on changer la hauteur et la durée du bip sonore produit par l'instruction BEEP :

☐ Non, seule l'instruction SOUND le permet.
☐ Oui, à la condition de faire suivre l'instruction BEEP de l'indication de hauteur et de durée du bip sonore (les valeurs de 800 hertz, 1/4 seconde étant fixées par défaut).

2. A quoi correspond la hauteur de la note jouée avec l'instruction SOUND 440, 4.4 ?

☐ à un DO.
☐ à un LA.

3. A quoi correspond la durée de la note jouée avec l'instruction SOUND 523, 8.8 ?

☐ à une noire pointée jouée sur un tempo de 180.
☐ à une noire jouée sur un tempo de 120.
☐ à une croche jouée sur un tempo de 60.

4. Comment obtient-on un silence avec l'instruction SOUND ?

☐ en programmant une note de fréquence suffisamment aigüe pour que l'oreille ne l'entende pas.
☐ en intercalant une boucle FOR... TO de temporisation

5. Comment change-t-on d'octave avec l'instruction PLAY ?

☐ en employant la lettre-code O suivie du nouveau numéro d'octave.
☐ en employant les signes > ou < pour passer à l'octave immédiatement supérieure ou inférieure.

6. Qu'est-ce le mode legato ?

☐ C'est un mode musical qui consiste à ne jouer la note que pendant les 7/8e du temps qui lui est alloué.
☐ C'est un mode musical qui consiste à supprimer les temps morts entre les notes.

Réponses page 234

ANNEXE A

LE MODE GRAPHIQUE EGA

A l'heure où ces lignes sont écrites, le mode graphique EGA (de l'anglais Extended Graphic Adapter...) semble s'implanter comme nouveau standard graphique sur les machines de type PC/XT/AT ainsi que sur les nouveaux PS (pour Personal System...) d'IBM. Il suffit pour s'en convaincre de regarder autour de soi ; il n'est pas un constructeur qui ne propose sa propre carte graphique EGA multifonction et le moniteur couleurs qui lui est associé.

Si donc côté matériel tout est en place pour la mise en œuvre de ce nouveau mode graphique disposant de 640×350 pixels affichables en 16 couleurs parmi 64 (ce qui soit dit en passant nécessite une mémoire d'écran de 112 K...), qu'en est-il du côté logiciels et plus particulièrement du côté des langages.

Dans ce domaine, il faut bien avouer que ce n'est guère brillant !... A ce jour, il existe quelques langages supportant le mode EGA, à savoir l'Extended BASIC d'IBM (non compilable par le compilateur BASCOM 1.Ø...), le Quick BASIC de Microsoft et le turbo BASIC, le GWBASIC 3.2, de Borland. Or, comme un fait exprès, aucun de ces langages n'est à même de pouvoir sauvegarder ou recharger une image EGA dans la mémoire d'écran. Les développeurs de ces langages ont fait l'impasse totale sur ces deux points précis, ce qui nous amène à nous poser cette question fondamentale :

— *Quel intérêt peut présenter pour l'utilisateur un langage de programmation supportant le mode graphique EGA, s'il ne lui permet pas de sauvegarder le fruit de son travail ?*

Maintenant, n'allez pas croire que la situation soit meilleure du côté de la recopie des écrans EGA sur imprimante. L'utilitaire de recopie (comparable au fichier GRAPHICS des modes graphiques CGA...) n'est livré en standard ni avec les machines, ni avec les cartes graphiques et moniteurs, ni avec les imprimantes (tout du moins pas encore...). A vous de vous « débrouiller » du mieux que vous pouvez selon le matériel et les logiciels dont vous disposez.

Face à de telles difficultés, vous pourriez penser que la solution réside dans l'emploi des logiciels graphiques du commerce, qu'ils fonctionnent ou non dans un environnement WINDOWS ou autre. Là encore, déception !... Imaginez-vous que ces derniers sauvegardent et rechargent les images EGA sous forme compactée (de façon à prendre moins de place sur la disquette ou le disque dur...), ce qui rend leurs fichiers totalement incompatibles avec vos propres outils de développement. La joie quoi !...

Fort de ces quelques constatations, vous n'avez d'autre choix (tout du moins dans le moment présent...) que de développer vous-même vos propres routines de chargement et de sauvegarde des fichiers d'image EGA, ainsi qu'éventuellement l'utilitaire de recopie d'écran et à les incorporer au sein du langage de programmation qui est le vôtre sous la forme d'une librairie associée.

Quand aux autres modes graphiques en cours d'émergence (640×480, 800×600, 1280×960, 1024×1024 et les autres...), la situation est on ne peut plus claire. Il n'existe d'autre langage support que celui pour lequel vous allez développer (ou acheter...) une librairie de routines graphiques spécialisées. Un point, c'est tout !

ANNEXE B

CODE ASCII ÉTENDU DU CLAVIER

Pour mémoire, il est rappelé que le code ASCII étendu du clavier correspond à la valeur du second octet (le premier valant Ø) que délivre l'instruction **INKEY$** à la suite de la frappe d'une touche.

Touche	2nd octet décimal	2nd octet hexadécimal
ESC	Ø1	Ø1
Backspace	14	Ø E
Tab	15	Ø F
Return/Enter	28	1C
Ctrl	29	1D
Shift (gauche)	42	2A
Shift (droite)	54	36
PrtSc	55	37
Alt	56	38
Caps Lock	58	3A
F1 à F1Ø	59 à 68	3B à 44
Num Lock	69	45
Scroll Lock	7Ø	46
Home	71	47
↑	72	48
Pg Up	73	49
−	74	4A
←	75	4B
5	76	4C
→	77	4D
+	78	4E
End	79	4F
↓	8Ø	5Ø
Pg Dn	81	51
Ins	82	52
Del	83	53

ÉPILOGUE

Vous voici arrivés au terme des 12 chapitres de cet ouvrage (j'en entends certains qui poussent un ouf de soulagement...). Si vous les avez parcouru dans l'ordre et si vous avez effectué les différents exercices de programmation qui vous étaient proposés, vous n'avez pas été sans remarquer la quasi-impossibilité d'obtenir des programmes graphiques suffisamment performants lorsqu'on ne dispose que du Basic Microsoft.

Pourquoi, cela ?... Tout d'abord, parce que le Basic Microsoft (BASICA sur IBM-PC, GWBASIC ou similaire sur compatibles) est un langage interprété relativement lent d'exécution et, si ses peformances sont encore acceptables dans le domaine de la gestion de fichiers, elles sont totalement inadaptées au mode graphique.

La solution ?... Elle consiste dans un premier temps à compiler le programme Basic avec un compilateur de type BASCOM ou similaire. Le gain en vitesse d'exécution que l'on obtient sur une application graphique est généralement de l'ordre d'un facteur 8. Si cela ne suffit pas, il vous faut alors faire appel à d'autres langages plus performants sur le plan vitesse, tels que le TURBO PASCAL, le FORTH, le langage C ou carrément l'assembleur, pour ne citer que les plus en vogue dans le moment présent.

La seconde raison qui ralentit la vitesse d'exécution des programmes graphiques est la pauvreté du Basic Microsoft en instructions spécifiquement graphiques. Pour ne citer que l'exemple du logiciel d'application GRAFELEC développé au chapitre 11, son exécution serait infiniment plus rapide (et sa programmation facilitée...) s'il existait une instruction DRAW capable de dessiner en mode complémentaire, ainsi qu'une instruction PIXEL précisant le nombre de pixels allumés ou éteints du décor de fond situés en dessous d'une forme graphique. Et ceci ne concerne qu'une toute petite application choisie parmi des milliers d'autres...

Or, ce qui vient d'être énoncé pour le Basic Microsoft est également vrai pour les autres langages évolués. Ne croyez pas que le simple fait de choisir un nouveau langage suffit à résoudre tous vos problèmes. S'il est vrai qu'un nouveau langage peut être plus rapide qu'un autre, il se peut aussi que l'absence de certaines instructions graphiques spécifiques à une fonction particulière vous oblige à développer plusieurs sous-programmes dont le temps d'exécution grève le bilan vitesse de l'ensemble au point que le résultat final est moins bon qu'avec un bon vieux Basic compilé.

Que conclure de tout ceci ?... C'est que si le graphisme vous plaît et si vous souhaitez aller plus loin dans cette voie, il vous faut à tout prix abandonner le langage Basic pour un autre car, à terme, celui-ci ne peut que vous mener à une impasse.

Tout ceci est bien beau, me répondrez-vous, mais lequel choisir ?... C'est précisément là que réside toute la difficulté. S'il existait un langage idéal, cela se saurait et il n'y aurait pas cette prolifération de langages nouveaux qui apparaissent au fil des années, le dernier arrivé releguant au rang d'antiquité le précédent langage encore considéré comme le « nec plus ultra » en la matière quelques semaines auparavant.

En ce domaine, votre choix ne peut être que personnel. Tout au plus pour vous guider puis-je vous dire qu'à ce jour (mais cela est également sujet à modification...), les programmeurs professionnels utilisent à part égale le langage C ou l'assembleur. Personnellement, j'ai un petit faible pour l'assembleur car il me permet, au-delà de la grande vitesse d'exécution qui est la sienne, de redéfinir de nouvelles routines graphiques lorsque celles existantes sont inappropriées ou tout simplement boguées. Mais, comme je l'ai dit plus haut, ceci n'est qu'un choix personnel...

Le graphiste de service

Solution des exercices sur le chapitre 12

1. Non, seule l'instruction SOUND le permet

2. à un LA.

3. à une noire pointée jouée sur un tempo de 180 OU à une noire jouée sur un tempo de 120 OU à une croche jouée sur un tempo de 60.

4. en programmant une note de fréquence suffisamment aigüe pour que l'oreille ne l'entende pas.

5. en employant la lettre-code O suivie du nouveau numéro d'octave OU en employant les signes > ou < pour passer à l'octave immédiatement supérieure oü inférieure.

6. C'est un mode musical qui consiste à supprimer les temps morts entre les notes.

INDEX DES MOTS-CLES

B

BEEP, 219
BLOAD, 167
BSAVE, 168

C

Chaîne de caractères, 49
CIRCLE, 27
COLOR, 8,9

D

DELETE, 115
DRAW, 97

F

FUSFORM, 110

G

GET, 141
GRAPHICS, 169

I

INSERT, 114

L

LINE, 15
LINE option B, 24
LINE option F, 40
LIST, 112
LOAD, 113

M

MOUSE, 175

P

PAINT, 41, 49
PLAY, 223
POINT, 131
PRESET, 12
PSET, 11
PUT, 142
PUT option OR, 162
PUT option PRESET, 171, 185
PUT option PSET, 165

R

RENAME, 113

S

SAVE, 114
SCREEN, 6, 158
SOUND, 220
STEP, 20
SWAP, 115
SYSTEM, 169

W

WINDOW, 155

TABLE DES MATIÈRES

Chapitre I – Modes graphiques à moyenne et haute résolution **5**
Sélection des modes graphiques 5
Choix de la couleur de fond 7
Choix de la couleur d'encre 9
Allumage et extinction d'un point 10
Dessin d'un trait 15
Les coordonnées exprimées en valeurs relatives 19

Chapitre II. – Les figures géométriques **23**
Rectangles et boîtes 23
Cercles 25
Ellipses 27
Arcs de cercle et secteurs circulaires 30
Polygones 33
Etoiles 35

Chapitre III. – Coloriages des graphiques **39**
Coloriage avec l'option F 39
Coloriage avec l'instruction PAINT 41
Des couleurs supplémentaires avec LINE 46
Des couleurs en supplément avec PAINT 49
Les couleurs en noir et blanc 52
Coloriage en haute résolution 55

Chapitre IV. – Représentation graphique des données **59**
Analyse du problème posé par la représentation graphique des données .. 59
Elaboration du logiciel DATAGRAPH 61
Ecran du menu principal 65
Ecran d'entrée des données 68
Sous-programme d'INPUT contrôlé 71
Ecran de sauvegarde et de chargement des données 74
Ecran de sélection d'un programme 76

Chapitre V. – Les diagrammes de gestion **79**
Les diagrammes à traits continus 79
Les diagrammes à aires superposées 84
Les diagrammes à colonnes 87
Les diagrammes à colonnes empilées 90
Les diagrammes à secteurs circulaires 92

Chapitre VI. – Les formes graphiques **97**
Le sous-langage DRAW 97
Règles de codage d'une forme graphique 104
Sauvegarde et chargement d'une forme graphique 106
Constitution d'une bibliothèque de formes 108
FUSFORM : Un gestionnaire de fichiers de formes 109

Chapitre VII. — **Les caractères graphiques** **117**
Affichage de caractères au format 80 colonnes 117
Définition d'une police de caractères graphiques 118
Sous-programme d'écriture en page graphique 125
Le retour des accents 128
Comment obtenir de grosses lettres 131
Les textes à lecture verticale 133

Chapitre VIII. — **L'animation graphique** **137**
Comment créer le mouvement 137
Conservation du décor de fond 140
Les instructions GET et PUT 141
Le contrôle des déplacements à partir du clavier 143
Séquence d'animation nécessitant plusieurs formes graphiques 146
Changements alternés de plans d'animation 147
Entrée et sortie progressive de l'écran 149

Chapitre IX. — **Traitement des images** **155**
Redéfinition des limites d'écran 155
L'effet de zoom que procure l'instruction WINDOW 155
Une véritable loupe électronique 160
Découpage et collage 163
L'effet miroir 165
Chargement et sauvegarde d'une image 167
Recopie des écrans graphiques 169

Chapitre X. — **La souris microsoft** **175**
Activation de la souris 175
Fonctions de programmation 178
Changement de curseur mobile 180
Dessin à main levée 183
Choix d'options dans un menu 185
Menus déroulants 187
Déplacement d'icônes sur l'écran 189

Chapitre XI. — **Application graphique : Le logiciel GRAFELEC de dessin de schémas d'électronique** **193**
Inventaire des ressources graphiques 193
Analyse d'une application 194
Conception du menu du logiciel GRAFELEC 195
Elaboration d'une grille de dessin 197
Constitution de la bibliothèque de symboles 198
Sélection et mise en place des symboles 203
Rotations et sorties d'écran 205
Le problème des points de connexion 207
Le logiciel GRAFELEC 210

Chapitre XII. — **Bruitage et musique** **219**
Bruitage d'un programme 219
Hauteur et durée des notes 220
Transcription d'une partition musicale 222
Le sous-langage musical PLAY 223
Hey Jude 229

Le mode graphique EGA 235
Code ASCII étendu du clavier 236

EDITIONS RADIO

Pratique des IBM PC
et compatibles graphisme et son
volume III
J.-C. Fantou

Service lecteurs

(à retourner à Éditions Radio, 189, rue Saint-Jacques, 75005 Paris)

Pour nous permettre de vous proposer des ouvrages toujours meilleurs, nous souhaiterions recevoir vos critiques, appréciations et suggestions sur le présent livre :

____

____

____

____

Quels sont les ouvrages (thème, sujet, niveau) que vous souhaiteriez voir publier par notre société ?

____

____

____

____

Nous vous remercions de votre confiance et de votre coopération.

Éditions Radio

Je désire recevoir gratuitement et sans engagement (mettre une croix dans la case) :

☐ Votre catalogue général (Electronique professionnelle et grand public, Informatique, Hi-Fi, Vidéo)

Nom : ____________________ Prénom : ____________________

Adresse : ____________________________________

____

Secteur d'activité et fonction : ________________________

____

CENTRES D'INTÉRÊTS

☐ Electronique professionnelle
☐ Electronique de loisirs
☐ Vidéo
☐ Hifi, CB...
☐ Micro-informatique professionnelle
☐ Micro-informatique de loisirs
☐ Autres : ....
....

Voir au dos "Correspondance Auteurs"

Pratique des IBM PC
et compatibles graphisme et son
volume III
J.-C. Fantou

EDITIONS RADIO

Service lecteurs

Correspondance auteurs

(à retourner à Éditions Radio, 189, rue Saint-Jacques, 75005 Paris)

Pour toute demande d'éclaircissements techniques relatifs à ce livre, formulez ci-dessous vos questions, avec le maximum de précisions :

Nom : ______________________ Prénom : ______________________

Adresse : ______________________

Egalement aux Editions Radio :

- Pratique des MS-DOS et PC-DOS
- 8088 et ses périphériques
- 80286 Assembleur IBM AT
- Pratique des IBM PC Vol 1

ÉDITIONS RADIO

PRATIQUE DES IBM PC et compatibles
GRAPHISME et SON

Simple de compréhension et pratique d'emploi, cet ouvrage vous propose des méthodes et des programmes de dessin fonctionnant indifféremment sur toute machine de type PC, XT, AT et compatible. Vous y découvrirez notamment comment :

- créer vos propres graphiques de gestion (en colonnes, en camemberts, etc.),
- concevoir et animer des formes graphiques,
- redéfinir vos propres polices de caractères,
- élaborer des images en couleurs (avec zoom, fenêtrage, etc.),
- programmer la souris Microsoft,
- développer de A à Z un utilitaire de DAO,
- utiliser les possibilités sonores de votre ordinateur pour agrémenter vos logiciels,

...et bien d'autres choses encore !

Accessible à tous, même aux débutants, ce livre vous permettra de maîtriser rapidement le graphisme sur PC.

9 782709 110112
ISBN 2 7091 1011 3
Code 41

ÉDITIONS RADIO

F 230 /87/7